AD-A040 274 COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON D C F/G 15/6
THE CCTC QUICK-REACTING GENERAL WAR GAMING SYSTEM (QUICK). PROG--ETC(U)
APR 77

UNCLASSIFIED CCTC-CSM-MM-9-74-VOL-4-PT NL

1 OF 2
AD
A040274

DEFENSE COMMUNICATIONS AGENCY

COMMAND AND CONTROL
TECHNICAL CENTER

WASHINGTON, D.C. 20301

12

B.S.

IN REPLY REFER TO: C314

11 14 April 1977

12 140p.

ADA040274

TO: RECIPIENTS

SUBJECT: Change 2 to Program Maintenance Manual CSM MM 9-74, Volume IV, Sortie Generation Subsystem

1. Insert the enclosed change pages and destroy the replaced pages according to applicable security regulations.

2. A list of Effective Pages to verify the accuracy of this manual is enclosed. This list should be inserted before the title page.

3. When this change has been posted, make an entry in the Record of Changes.

FOR THE DIRECTOR

146 Enclosures
Change 2 pages

J. DOUGLAS POTTER
Asst to the Director
for Administration

6 The CCTC Quick-Reacting General War Gaming System (QUICK). Program Maintenance Manual. Volume IV. Sortie Generation Subsystem.

DDC
RECEIVED
JUN 2 1977
A

14 CCTC-CSM-MM-9-74-Vol-4-Pt-1-Ch-2

AD NO.
DDC FILE COPY

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

AMERICAN REVOLUTION BICENTENNIAL 1776-1976

409658

1/B

EFFECTIVE PAGES - JUNE 1976

This list is used to verify the accuracy of CSM MM 9-74 Volume IV after change 2 pages have been inserted. Original pages are indicated by the letter 0, change 1 pages by the numeral 1, and change 2 pages by the numeral 2.

Page No.	Change No.	Page No.	Change No.
Front Cover, Part I	0	148-151	2
Title Page	0	152-153	0
ii	1	154-155	2
iii-iv	0	156-160	0
v-vi	1	161-162	2
vii-x	0	163-184	0
xi	1	185-186	2
xii	0	187-188	0
1-9	0	189	2
10	2	190-332	0
11-18	0	733-736	1
19-29	2	Front Cover, Part II	1
29.1-29.2	2	Title Page	0
30-32	2	ii	1
33-54	0	iii	0
55-56	2	iv-vi	1
57-59	0	vii	0
60	2	viii	2
61-65	0	ix	0
66-68	2	x	2
69-71	0	xi	1
72	2	xii	0
73-74	0	333-338	0
75-84	2	339	2
84.1-84.2	2	340	0
85	2	341	2
86-89	0	342-350	0
90	2	351	2
91-111	0	352	0
112	2	353	2
113	0	354-355	0
114	2	356-357	2
115	0	357.1-357.2	2
116	2	358-360	2
117-139	0	361-371	0
140-142	2	372	2
142.1-142.2	2	373-392	0
143-145	2		
145.1-145.2	2		
146-147	2		
147.1-147.2	2		

EFFECTIVE PAGES - Continued

Page No.	Change No.
393-394	2
394.1-394.2	2
395-399	2
400-425	0
426	2
427-428	0
429	2
430-433	0
434	2
435-444	0
445	2
446-449	0
450	2
451-569	0
570	2
571-614	0
615	1
616-619	2
619.1-619.2	2
620	2
621	0
622	2
623-627	1
628-644	0
645-647	2
648-651	1
651.1-651.2	2
652-653	2
654-655	0
656	2
657-658	0
659	2
660-732	0
733-736	1

For those weapon groups with a MIRV capability, the data on the TMPALOC file are read into core. The program creates the individual booster assignments and outputs the strike information onto the ALOCGRP file. For MIRV groups, the data are organized as follows (see table 1). The index number of each target that receives the first reentry vehicle (RV) from a booster is set negative. The strikes are ordered such that targets which receive successive RVs from the same booster are listed in that order. (See also following section: Concept of Operation.)

2.4 Concept of Operation

Program ALOCOUT prepares the TMPALOC file on which information concerning target assignments is sorted by group. For those groups with a payload containing multiple independently-targetable reentry vehicles (MIRV), program FOOTPRNT performs further processing. The inclusion of a MIRV capability into the QUICK system is based upon the assumption that the MIRV weapons can be allocated to targets without regard to the "footprint" constraints. These constraints define the geographic area into which the ordered set of reentry vehicles from a single booster must be targeted.) This design approach considers that if a certain amount of extra or excess strikes are included in the allocation, the footprint constraints can be imposed later without the loss of payoff. Since imposition of the constraints may show that a certain number of strikes contained in the unconstrained allocation are not capable of inclusion in a feasible footprint, the extra strikes are added so that the final assignment contains the correct number of strikes.

This program prepares the ALOCGRP file for use by program POSTALOC. This file is very similar to the TMPALOC file. For those weapon groups with a MIRV capability, the data set on ALOCGRP differs from that on TMPALOC in the following ways:

a. The "extra" strikes have been removed

b. The index number (INDEXNO) of the target which receives the first reentry vehicle (RV) from each booster is set negative

c. The strikes are ordered such that:

- o Within each booster load (i.e., between minus signs) the strikes are ordered in order of their delivery by the missile
- o The booster loads are ordered by decreasing value (as defined by the sum of the relative damage values (RVAL) for all targets assigned to the booster).

Table 1. Format for MIRV Group Records on ALOCGRP File

ASSOCIATED COMMON	VARIABLE OR ARRAY	LENGTH	DESCRIPTION
STRKSUM	KGROUP	1	Group number
	NTSTRK	1	Total number of strikes for this group
	NCORR	1	Number of corridors for this group (=1)
STRKSUM	NSTRK	30	Number of strikes assigned to each corridor
RAIDATA	NT	1	Total number of targets assigned to group
	JGROUP	1	Group number
	JCORR	1	Corridor number (=0)
	INDEX	NT	Index numbers of targets (negative if first target assigned to booster)
	TGTLAT	NT	Target latitude (degrees)
	TGTLONG	NT	Target longitude (degrees)
	RVAL	NT	Relative value of strike
	DLAT	NT	Offset latitude (degrees)
	DLONG	NT	Offset longitude (degrees)
RAIDATA	LXLLFIX	(NT/36) +1	Fixed assignment indicator
--	LXIHOB	(NT/36) +1	Weapon height of burst indicator
C4	DESIG	NT	Target designator code
	TASK	NT	Target task and country owner codes
	CNTRYLOC	NT	Target country location code
C4	FLAG	NT	Target flag code
C5	ISAL	NT	Salvo number

Table 2. (Part 3 of 3)

INPUT FROM TMPALOC AND OUTPUT ON ALOCGRP

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
STRKSUM	KGROUP	Group number
	NTSTRK	Total number of strikes assigned
	NCORR	Number of penetration corridors used
	NSTRK(30)	Number of strikes assigned to each penetration corridor
	LSTRKSUM	Length of STRKSUM record
RAIDATA*	NT	Total number of strikes
	JGROUP	Group number
	JCORR	Penetration corridor
	INDEX(1500)	Target index number
	TGTLAT(1500)	Target latitude
	TGTLONG(1500)	Target longitude
	RVAL(1500)	Relative value for target
	DLAT(1500)	DGZ offset latitude (degrees)
	DLONG(1500)	DGZ offset longitude (degrees)
	LRAID	Length of /RAIDATA/ block to this point
	NTMAX	Maximum number of target assignments for one group
	LXLLFIX(41)	Fixed assignment indicator (1500 packed logical values)
C4	DESIG(1500)	Target designator code
	TASK(1500)	Target task/subtask and country owner codes
	CNTRYLOC(1500)	Target country location code
	FLAG(1500)	Target flag code
C5	ISAL(1500)	Salvo number

* This block is redefined for internal use - see internal common block /RAIDATA/ in table 3.

Table 3. Program FOOTPRNT Internal Common Blocks (Part 1 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
RAIDATA	NT	Total number of strikes
	JGROUP	Group number
	JCORR	Penetration corridor
	INDEX* (1500)	Target index number
	R(1500)	Distance from group centroid to DGZ (nautical miles)
	THETA(1500)	Launch azimuth of weapon from centroid to DGZ (radians)
	RVAL(1500)	Relative value for target
	IFOR(1500)	Forward pointer for booster assignments
	IBACK(1500)	Backward pointer for booster assignments
	LRAID	Length of /RAIDATA/ block to this point
	NTMAX	Maximum number of target assignments for one group
	LXLLFIX(41)	Fixed assignment indicator (1,500 packed logical values)
CONTROL	NV	Number of boosters in group
	NARV	Average number of targets per booster in initial assignment
	NEXTRA	Number of boosters with initial assignments containing (NARV + 1) reentry vehicles
	PEXTRA**	Fraction of total strikes that are excess strikes added by PREPALOC
	NPASS	Processing pass number

* Array IDUM, used for input/output temporary storage equivalenced to this array.

** From common block /EXCESS/ on BASFILE.

Table 3. (Part 2 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
CONTROL (cont.)	FRACLOOK	Fraction of next booster load for look-ahead
	MAXFOOT	Input parameter governing degree of effort expended in subroutine OPTBOOST
	DELAGE	Multiplier for AGE in potential target arrays
	PURGE	Fraction of targets in potential target arrays removed in BOOSTIN
	PN	Weighting factor for worth function
	EXTRAB*	Number of extra booster loads added in PREPALOC
	NOK	Actual number of correct strikes to be assigned
	IGSTART	First group to process
	IGEND	Last group to process
DSQUARE	CD2	Square of CROSSDWN } see common /RANGE/
	UD2	Square of UPDOWN } see common /RANGE/
	DEL2	Square of DELMIN } see common /RANGE/
	DZ2	Square of DZ } see common /VALPARM/
	VMIN	=VALF(DELMIN/DZ,TNZ) } see common /VALPARM/
EARTH	RADIUS	Radius of Earth (Nautical miles)
	DEGTORAD	Conversion factor for degrees to radius
	PI	Pi
	PIDIV2	Pi/2
FOOTIO	MAXRV	Maximum number of reentry vehicles allowed in one assignment
	ISYS	System identification number
	NTAR	Number of reentry vehicles currently assigned

* From common block /EXCESS/ on BASFILE.

Table 3. (Part 3 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
FOOTIO (cont.)	RIN(20)	Range to target (nautical miles)
	THIN(20)	Azimuth to target (radius)
	IFEAS	Number of targets that can be reached within fuel constraints
	DELRSTRT	Maximum additional flying distance allowed if first target in footprint is to be changed (nautical miles)
	DELRAFT(20)	Maximum additional flying distance allowed if new target is to be added after this target in footprint (nautical miles)
	FUELEFT	Fuel left after completion of weapon deliveries
FOOTSAVE	IFOTSAVE(20)	Potential target index of targets in first footprint
	NHITOLD	Number of targets in first footprint
	VHITOLD	Sum of RVALs for targets in first footprint
	IF2SAVE(20)	Potential target index of targets in second footprint
	N2SAVE	Number of targets in second footprint
INDEX	JINR	RAIDATA index of target to be entered into potential target arrays
	JINP	Potential target index of target to be entered
	JOUTR	RAIDATA index of target to be removed from potential target arrays
	JOUTP	Potential target index of target to be removed
	JSAVE(20)	Potential target index of targets entered by look-ahead

Table 3. (Part 4 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
INDEX (cont.)	NJSAVE	Number of targets entered by look-ahead
	JSAVOPT	Look-ahead flag
LOADATA	LOADOPT	Booster loading option
	NRVADD(1500)	Number of extra reentry vehicles added to this target
	NADDED	Total number of extra RVs added in a pass
	NTOADD	Number of RVs to be added to current footprint
	NONTAR(20)	Total number of RVs on each target in assignment
	NADDOLD	Number of extra RVs added in first pass
PARAMETR	MAXSYS	Maximum number of systems allowed in footprint parameter table
	IHNAME(40)	Hollerith name of MIRV system
	MINLOAD(40)	Minimum number of RVs per booster
	MAXLOAD(40)	Maximum number of RVs per booster
	DSPACE(40)	Minimum spacing (nautical miles) between consecutive DGZs in footprint
	THROWMAX(40)	Maximum distance between consecutive DGZs in footprint (nautical miles)
	MTYPE(40)	Footprint constraint functional form designator
	IDATA(40)	Index to footprint parameter data set
PERFORM	NASGN	Total number of targets assigned to boosters in current pass
	VALASGN	Sum of RVALs for all targets assigned in current pass
	TVAL	Sum of RVALs for all targets
	NOLD	Number of targets assigned in first pass

Table 3. (Part 5 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
PERFORM (cont.)	VALOLD	Sum of RVALs for targets assigned in first pass
POTENT	MAXPOT	Maximum number of potential targets
	MAXHIT	Maximum number of targets in hit list
	IPOT(50)	RAIDATA index of potential targets
	NHIT	Number of targets in hit list
	IHIT(20)	Hit list - potential target index
	TOFLY(20)	Distance (nautical miles) between successive targets in hit list
	NMISS	Number of targets in miss list
	MISS(50)	Miss list - potential target index
	NFREE	Number of available spaces in potential arrays
	IFREE(50)	Potential target index of available spaces
	NLOST	Number of "lost" targets
	LOST(50)	RAIDATA index of "lost" targets
	INVERSE(50)	Index to position in hit or miss list
	AGE(50)	Factor related to number of boosters processed while target remains in potential target arrays
RANGE	CROSSDWN	Ratio of downrange to crossrange distance
	UPDOWN	Ratio of downrange to uprange distance
	DELMIN	Minimum spacing between consecutive DGZs in a footprint
	DEFAULT	Minimum spacing allowed for computation
TSCRATCH	ISCR	Logical unit number for assignment data scratch file

Table 3. (Part 6 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
TSCRATCH (cont.)	ITABL	Logical unit number for footprint parameter data scratch file
VALPARM	DZ	Maximum distance between consecutive DGZs in footprint
	TNZ	Intercept for value line (determined by PN in /CONTROL/)
	SLZ	Slope of value line
WPNTGT	IPOTGT	Potential target index of target to be added or deleted from hit list
	JAFT	Potential target index of target after which new target is to be added in hit list
	JTGTD	RAIDATA index of target to be removed from a booster assignment
	NUMBOOST	Booster number currently being processed
C1	VAL(50)	Worth of target if added to footprint
	JAFTER(50)	Potential target index of target preceding new target in footprint
	VALFIRST(50)	Worth of making target first target in footprint
	COSTEFF(20)	Inverse of additional fuel needed to reach this target
	D(50,50)	Distance computation matrix
C2	MAXBOOST	Maximum number of boosters allowed in one group
	IBOOST(500)	RAIDATA index of first target assigned to booster
	NTB(500)	Number of targets assigned to booster
	ISTATUS(1500)	Target processing status
	NDEX(1500)	Temporary index storage

Table 3. (Part 7 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
C3	RP(50)	Range of target
	TP(50)	Azimuth to target
	RVALP(50)	Target relative value
	SINES(50)	Sine of azimuth
	COSINES(50)	Cosine of azimuth
	AVRP	Average range of all potential targets
	AVTHP	Average azimuth of all potential targets
	RHIT	Range to first target in footprint
	THIT	Azimuth to first target in footprint
	SINAV	Sine of AVTHP
	COSAV	Cosine of AVTHP
	SINHIT	Sine of THIT
	COSINHIT	Cosine of THIT
	THOLD	Azimuth used to compute entries in distance matrix
	COSINER(50)	Cosine of great arc from weapon group to target
	SINER(50)	Sine of the same angle as in COSINER
DEBUG	IOTA	Index to last entry in ICAMFROM array
	ICAMFROM(20)	Index of Hollerith names of subroutine calling sequence
PRINT	ICALL	Print request number
	IMUST	Error condition indicator
FLAG	NFLAG	Maximum number of print options
	LXIFLAG(3)	Active print indicator
	NC	Number of print requests
	IPRNT(60)	Print option number

Table 3. (Part 8 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
FLAG (cont.)	IFG(60)	First group to be printed
	IFP(60)	First pass to be printed
	IFB(60)	First booster to be printed
	ILG(60)	Last group to be printed
	ILP(60)	Last pass to be printed
	ILB(60)	Last booster to be printed
	MYPRT(60)	Mode by which print was requested (DEFAULT, INPUT, or REMOVED)
	IDUMP	Print number to abort run with memory dump

The following blocks contain the parameters which define the footprint constraints. The descriptions of subroutines TABLINPT and SETDATA contain more detailed information.

FOOTDATA (long-range system)	GAS(16)	Fuel available for footprinting
	RX(16,2)	Basic range extension coefficient
	RAXX(16,2)	Added range extension coefficient
	TOSSC1(16,2,16) TOSSC2(16,2,16)	Fuel consumption parameters
	TEONE(16,16) TETWO(16,16)	Fuel consumption exponents
	TDENOM(16)	Distance scaling factor
	RBASIC(16,2)	Basic maximum booster range
	RADD(16,2)	Added maximum booster range
	EONE(16) ETWO(16)	Downrange-crossrange ratio exponents
	DENOM	Distance scaling factor
	CONE(16,2) CTWO(16,2)	Downrange-crossrange ratio coefficients
	UE1(16) UE2(16)	Downrange-uprange ratio exponents

Table 3. (Part 9 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
FOOTDATA (cont.)	UC1(16) UC2(16)	Downrange-uprange ratio coefficients
	UDEN	Downrange-uprange distance scaling factor
	LLNGDAT	Length of this block
SHRTDAT (short-range system)	ALPHAZ(16) ALPHA1(16) ALPHA2(16)	Fuel consumption parameters
	BETAZ(16) BETA1(16) BETA2(16)	Fuel load parameters
	MAXRBOST(16)	Maximum booster range
	GTWO GONE GZERO	Downrange-crossrange ration parameters
	DONE DZERO	Downrange-uprange ratio parameters
	LSHTDAT	Length of this block
PENADD (additions for penetration aids)	TOTFUEL	Total fuel available for spacing, release, and footprinting
	SRFC1(16) SRFC2(16)	Spacing and release fuel coefficients
	SRFEXP1(16) SRFEXP2(16)	Spacing and release fuel exponents
	SRFDEN	Distance scaling factor
	LPENDAT	Length of this block
CSYS4	A0(16) A1(16) A2(16)	Fuel consumption parameters

Table 3. (Part 10 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTIONS
CSYS4 (cont.)	B0(16) B1(16) B2(16)	Fuel load parameters
	RANGEB(16,2)	Basic maximum booster range
	BRADD(16,2)	Added maximum booster range
	CR2 CR1 CR0 CRDEN	Downrange-crossrange ratio parameters
	UD2 UD1 UD0	Downrange-uprange ratio parameters
	LDSYS4	Length of this block
WAROUT	IWARFL	Logical unit number for war gaming print output
HDATA		List of values for hollerith testing
	HDEFAULT	6HDEFAUL
	HIGSTART	6HIGSTAR
	HI8	2HI8
	HIGEND	5HIGEND
	HLOADOPT	6HLOADOP
	HA8	2HA6
	HFREE	4HFREE
	HMAXFOOT	6HMAXFOO
	HFRACLOO	6HFRACLO
	HF8x2	4HF8.2
	HDELAGE	6HDELAGE
	HPN	2HPN

Table 3. (Part 11 of 11)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
HDATA (cont.)	HPURGE	5HPURGE
	HIDUMP	5HIDUMP
	HPRINT	5HPRINT
	HNOPRINT	6HNOPRIN
	HXINPUT	6H INPUT
	HSCRATCH	6HSCRATC
	HPOSITIV	6HPOSITI
	HREMOVED	6HREMOVE
	HNEGATIV	6HNEGATI
	HBLANKXX	6H
	HXNMXLBX	6H NM/LB
	HXDRXCRX	6H DR/CR
	HPOUNDSX	6HPOUNDS
	HBASFILE	6HBASFIL
	HMAYX73	6HNOV 73
	HTMPALOC	6HTMPALO
	HALOCGRP	6HALOCGR
	HEOT	3HEOT
	HNOTXUSE	6HUNUSED

THIS PAGE INTENTIONALLY LEFT BLANK

Table 4. List of Initial Settings of Variables**
(Part 1 of 2)

An asterisk flags variables whose values are changed during processing.

VARIABLE	BLOCK	INITIAL VALUE	REMARKS
AGE*	POTENT	0.0	Length of time target has remained in potential target list
AZDIFF		.01	Used by subroutine REVAL to determine necessity of recomputing distance matrix
AZOLD*		10^{-9}	Used by subroutine FOOTEST to determine necessity of recomputing fuel consumption parameters
DEFAULT	RANGE	1.0	Minimum spacing of DGZs required for computation
DEGTORAD	*EARTH*	*.0174532*	*Conversion factor-degrees to radians*
EPSILON*		10^{-9}	Same use as AZOLD
IERR*		0	Error counter in subroutine FOOTEST
ILASTL*		0	Used by subroutine SETDATA to determine if new footprint data are required
ILASTP*		0	
ILASTS*		0	
IMUST*	PRINT	0	Error condition indicator
ISCR	TSCRATCH	25	Scratch file logical unit (assignment data)
ITABL	TSCRATCH	26	Footprint data file logical unit
LLNGDAT	FOOTDATA	1890	Number of words in long-range system data set
LPENDAT	PENADD	1956	Number of words in penetration aids system data set
LRAID	RAIDATA	9003	Length of /RAIDATA/ block

**This list does not include the default values of the user-input parameters which are described in subroutine *RDCARDF*.

Table 4. (Part 2 of 2)

VARIABLE	BLOCK	INITIAL VALUE	REMARKS
LSHTDAT	SHRTDAT	117	Length of short-range system data set
LDSYS4	CSYS4	167	Length of Type-4 system data set
LSTRKSUM	STRKSUM	33	Length of /STRKSUM/ block
MAXBOOST	C2	500	Maximum number of boosters per group
MAXHIT	POTENT	20	Maximum number of RVs in one footprint
MAXPOT	POTENT	50	Maximum number of entries in potential target list
MAXRV	FOOTIO	20	Maximum number of RVs in footprint that can be tested
MAXSYS	PARAMETR	40	Maximum number of systems that can be considered in one run
NFLAG	FLAG	100	Maximum number of print options
NTMAX	RAIDATA	1500	Maximum number of targets per group
PDIFF		.001	Used by subroutine FOOTEST to determine necessity of fuel parameter recomputation
PI	EARTH	3.1415927	
PIDIV2	EARTH	1.5707963	
RADIUS	EARTH	3440.068483	Nautical miles
THOLD*	C3	10^{+37}	Azimuth used for distance matrix
XOLD		10^{-9}	Same as AZOLD

2.6 Program FOOTPRNT

PURPOSE: This is the main program. It acts as a control driver for the rest of the subroutines. It is the interface subroutine between this program and the remainder of the QUICK system.

ENTRY POINTS: FOOTPRNT

FORMAL PARAMETERS: None

COMMON BLOCKS: C1, C2, C3, C4, C5, CONTROL, DEBUG, DSQUARE, EARTH, FILABEL, FILES, FOOTIO, FOOTSAVE, HDATA, IFTPRNT, INDEX, ITP, LOADATA, MASTER, MYIDENT, MYLABEL, NOPRINT, PARAMETR, PERFORM, POTENT, PRINT, RAIDATA, RANGE, STRKSUM, TSCRATCH, TWORD, VALPARM, WPNGRPX, WPNTGT

SUBROUTINES CALLED: BOOSTIN, BOOSTOUT, GLOG, GOPRINT, INITAPE, INITRANS, INITASGN, LREORDER, NEWCOOR, OPTBOOST, ORDER, PRINTSET, RDARRAY, RDCARDF, RDWORD, REMOVE, REORDER, SETDATA, SETREAD, SETWRITE, SKIP, SLOG, TERMTAPE, TRANSFER, WRARRAY, VALF

CALLED BY: Operating System; this is a main program

Method:

The functioning of program FOOTPRNT can be divided into five parts; the flowchart and the following description are similarly divided. The parts are: the initialization of the program control variables, reading the strike data and determining the groups with the MIRV capability, setting the control data for each individual MIRV group, generating the footprints for each booster in the group, and finally selecting, formatting, and writing the final plan. The majority of the file reading and writing is accomplished in this program and the specific cases are discussed in later paragraphs.

Part I - The Initialization of Control Variables

The functioning of this part of the program is quite straightforward logically. The program begins by calling subroutine INITAPE to initialize the filehandler. Subroutine RDCARDF is then called to read and interpret the user-input parameters. These parameters include the print requests, program control variables, and footprint parameter data tables. The use

2.6.3 Subroutine BOOSTIN

PURPOSE: This routine determines the set of potential targets for each booster and computes detailed intertarget parameters for all targets in the potential target list.

ENTRY POINTS: BOOSTIN

FORMAL PARAMETERS: None

COMMON BLOCKS: C1, C2, C3, C4, CONTROL, DEBUG, DSQUARE, EARTH, FOOTIO, INDEX, PARAMETR, POTENT, PRINT, RAIDATA, RANGE, WPNTGT, VALPARM

SUBROUTINES CALLED: CRSTODWN, GOPRINT, INPOT, ORDER, OUTPOT, UPTODOWN, VALF

CALLED BY: FOOTPRNT

Method:

This routine is called once each pass for each booster. Its purpose is to set up the potential target arrays for the booster. Its functions are:

a. Remove targets from the potential target arrays

b. Search for unassigned targets in the neighboring geographic area and place them in potential target arrays

c. Enter targets currently assigned to the booster into the potential target arrays

d. Compute intertarget distance matrix

e. Determine worth of maintaining each target in the array for the next booster processed

f. Compute the worth of starting the footprint with each target.

Processing begins with the search for "lost" targets. These are targets which are currently unassigned to a booster and not in the potential target arrays. This search is done only on the second pass since the

initial assignment generated by subroutine INITASGN contains every target. The geographic area to be searched is determined by targets currently assigned to the booster and also the targets assigned to the next booster to be processed. The backward pointer (IBACK) of the first target in each footprint is set to the RAIDATA index of the target with the largest launch azimuth in the footprint. Thus BOOSTIN uses this value of IBACK for each of the two boosters to determine the area of the RAIDATA list to investigate. Any targets in this area which are neither assigned nor in the potential target arrays (i.e., ISTATUS = -2) are placed in the lost target list (LOST) and ISTATUS is set to -1.

The routine now determines which targets in the potential target arrays should be removed to make room for the targets to be entered. The worth of maintaining a target in the potential list is always stored in the diagonal elements of the distance matrix D. The worth of maintaining the target whose POTENT index is J is saved in D(J,J). The number of targets to be dropped is determined by the input parameter PURGE. First the routine computes the number of targets in the POTENT list which were not entered in the list by the look-ahead feature of subroutine OPTBOOST. If this number is less than the average number of targets per booster, no targets are removed. Otherwise the routine omits the fraction, PURGE, of these targets. The targets are omitted in order of increasing worth. If this fraction removed does not leave sufficient room for the current assignment, targets are removed singly until there is sufficient room. The routine then enters the current booster assignment into the list. Finally, as many of the lost targets as possible are entered.

Two sets of intertarget parameters are now computed. First, the intertarget distance matrix is computed. The entries in this array, D, are defined as follows, for targets whose POTENT indices are i and j:

$$D(i,j) = \begin{cases} \text{Square of actual downrange distance from} & \\ \text{target } i \text{ to target } j \text{ (this quantity is set} & i < j \\ \text{negative if } j \text{ is uprange of } i\text{)} & \\ \text{Worth of maintaining target in potential list} & i = j \\ \text{Square of actual crossrange distance from} & i > j \\ \text{target } i \text{ to target } j & \end{cases}$$

The off diagonal terms are computed first by spherical and plane geometry. For this calculation, the downrange axis is defined to have the average launch azimuth of all the targets in the list.

The diagonal terms, the worth of maintaining the target in the list, are computed next. In order to keep targets in the arrays for at least two boosters, a target that has just been entered is given an artifically high value.

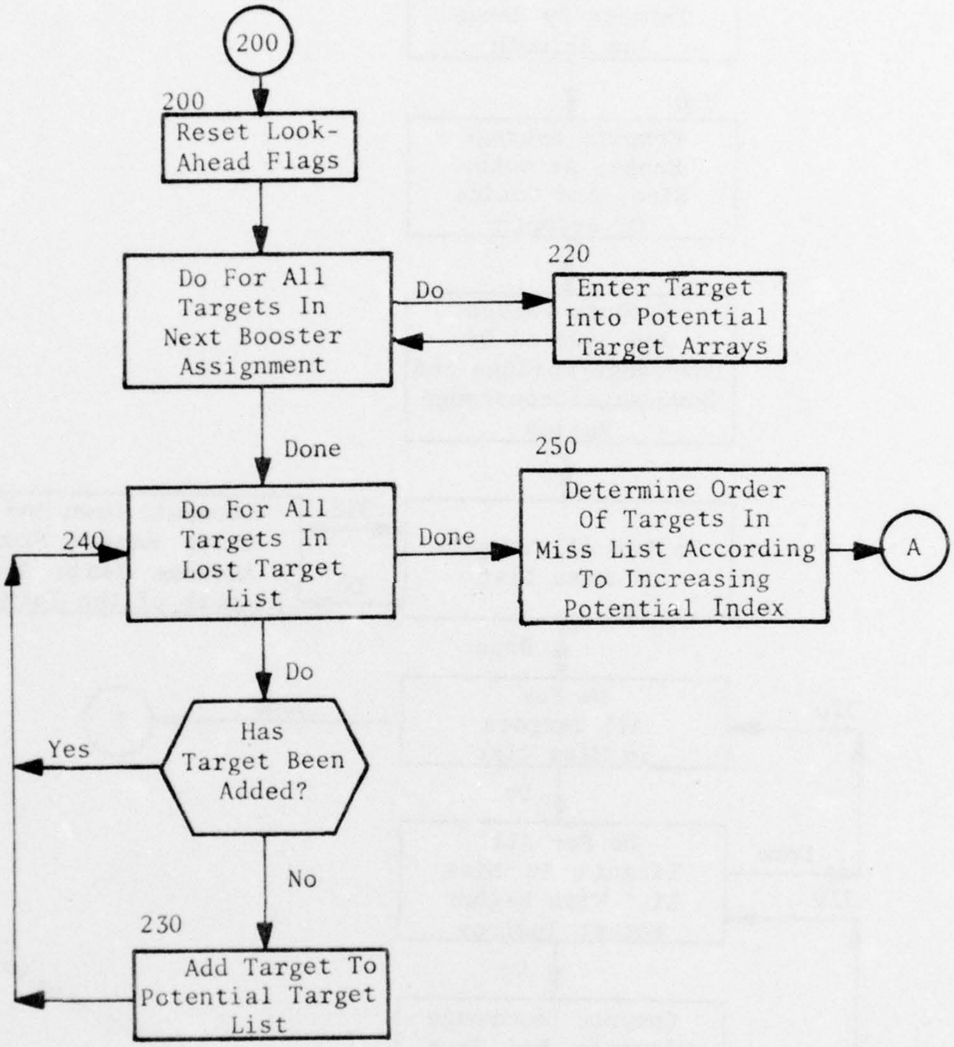


Figure 11. (Part 2 of 4)

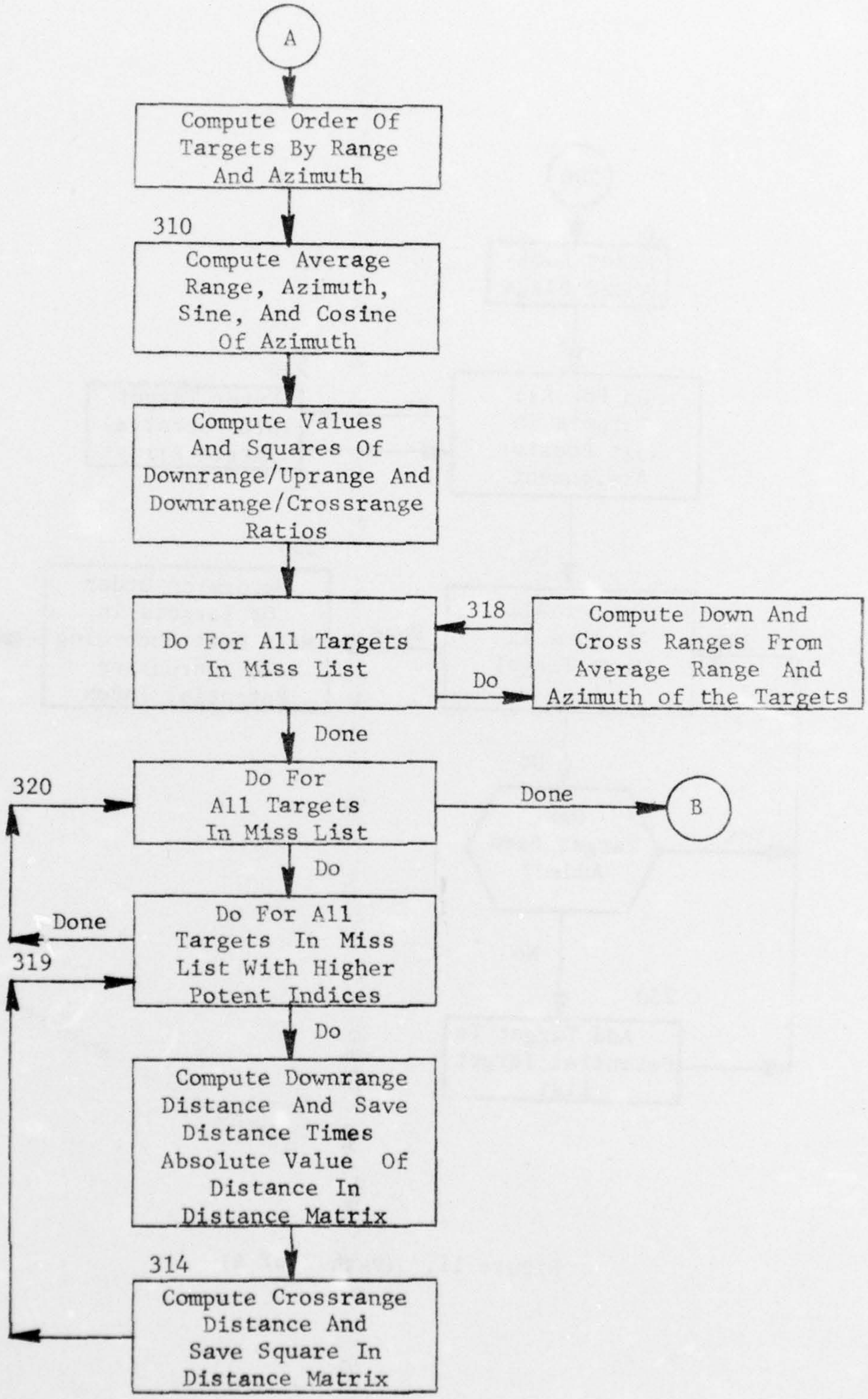


Figure 11. (Part 3 of 4)

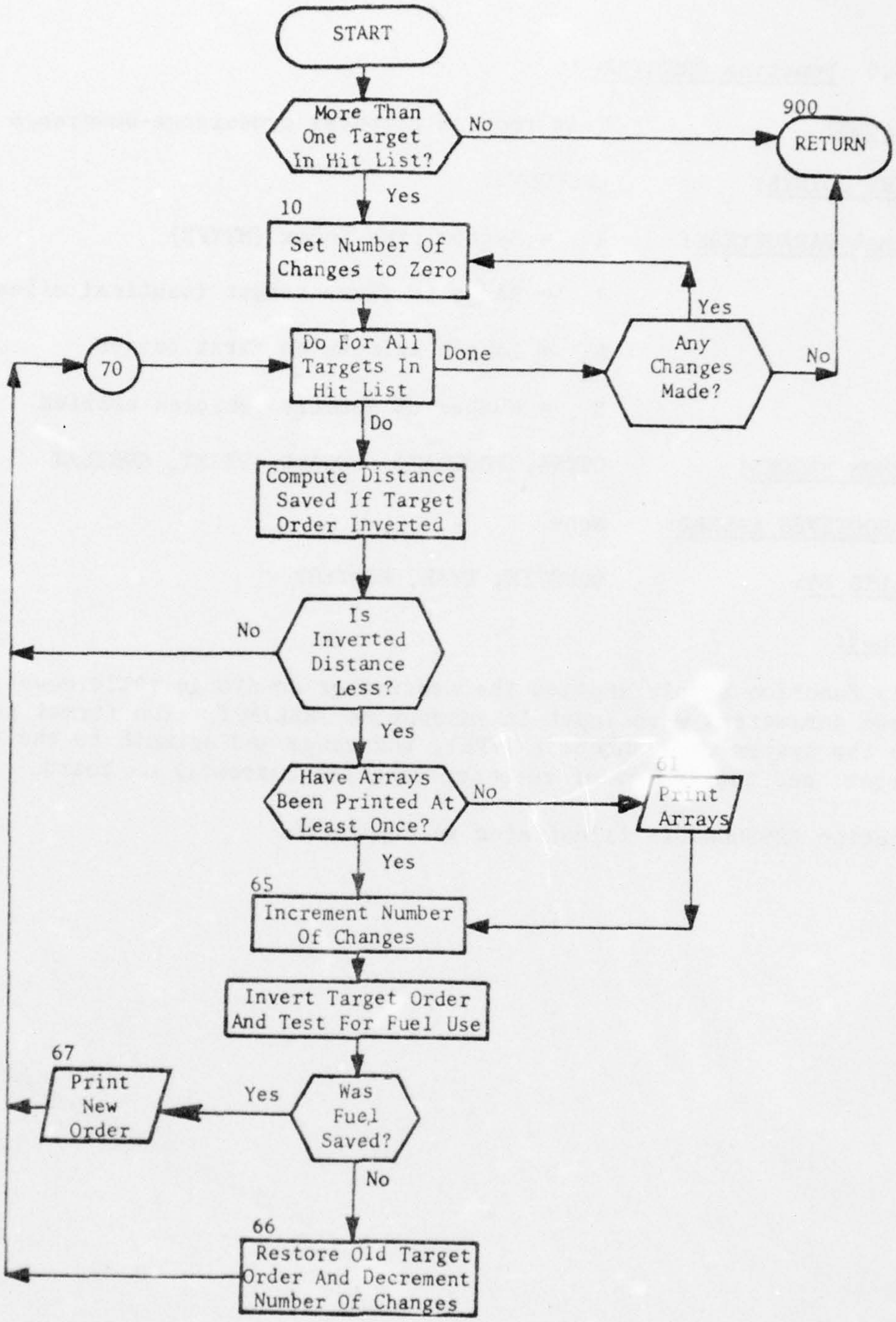


Figure 13. Subroutine CHKSEQ

2.6.6 Function CRSTODWN

PURPOSE: This routine computes crossrange-downrange ratios.

ENTRY POINTS: CRSTODWN

FORMAL PARAMETERS:

I = System type index (MTYPE)

R = Range to first target (nautical miles)

AZ = Launch azimuth to first target

N = Number of reentry vehicles carried

COMMON BLOCKS: CSYS4, FOOTDATA, PENADD, PRINT, SHRTDAT

SUBROUTINES CALLED: None

CALLED BY: BOOSTIN, EVAL, FOOTEST

Method:

This function simply applies the crossrange-downrange ratio equations whose parameters were input in subroutine TABLINPT. The formal parameters are the system type number (MTYPE), the range and azimuth to the first target, and the number of re-entry vehicles currently on board.

Function CRSTODWN is illustrated in figure 14.

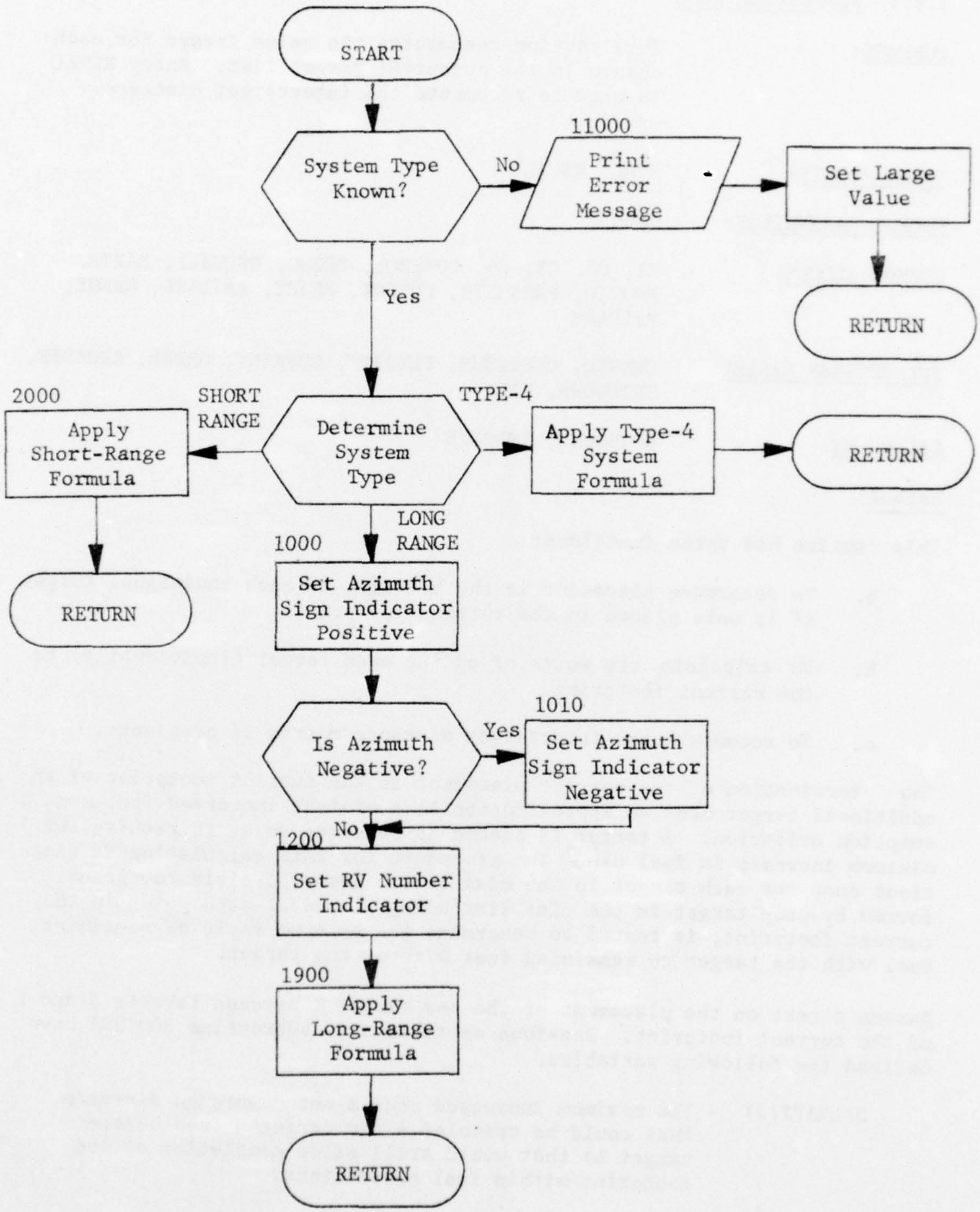


Figure 14. Function CRSTODWN

2.6.7 Subroutine EVAL

PURPOSE: This routine recomputes the value arrays for each change in the potential target list. Entry REVAL is used to recompute the intertarget distance matrix.

ENTRY POINTS: EVAL, REVAL

FORMAL PARAMETERS: None

COMMON BLOCKS: C1, C2, C3, C4, CONTROL, DEBUG, DSQUARE, EARTH, FOOTIO, PARAMETR, POTENT, PRINT, RAIDATA, RANGE, VALPARM

SUBROUTINES CALLED: CHKSEQ, CRSTODWN, FLYDIST, GOPRINT, ORDER, REORDER, UPTODOWN, VALF

CALLED BY: OPTBOOST, IMPROVE

Method:

This routine has three functions:

a. To determine placement in the hit list of each unassigned target if it were placed in the current footprint.

b. To calculate the worth of adding each target (individually) to the current footprint.

c. To recompute the intertarget distance matrix if necessary.

The determination of the correct placement in the current footprint of an additional target uses an approximation to a minimal increased fuel consumption criterion. A target is placed in sequence so as to require the minimum increase in fuel use. The procedure for this calculation is exercised once for each target in the miss list. Every possible footprint, formed by each target in the miss list being placed at each point in the current footprint, is tested to determine the maximum ratio of remaining fuel with the target to remaining fuel without the target.

Assume a test on the placement of the new target K between targets J and L of the current footprint. Previous operations in subroutine FOOTEST have defined the following variables.

DELRAFT(J) - The maximum increased equivalent downrange distance that could be traveled after target J (and before target L) that would still allow completion of the footprint within fuel constraints.

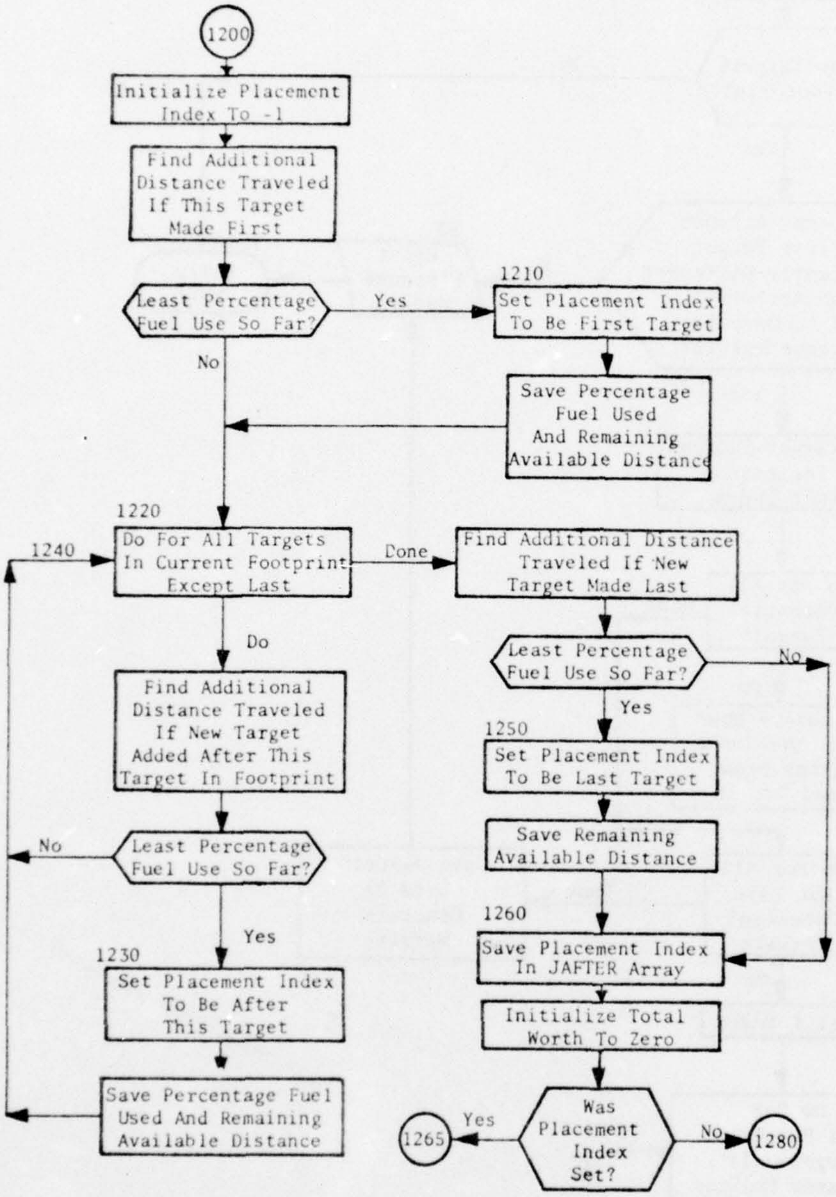


Figure 15. (Part 2 of 3)
Part II: Determination of Target Sequence

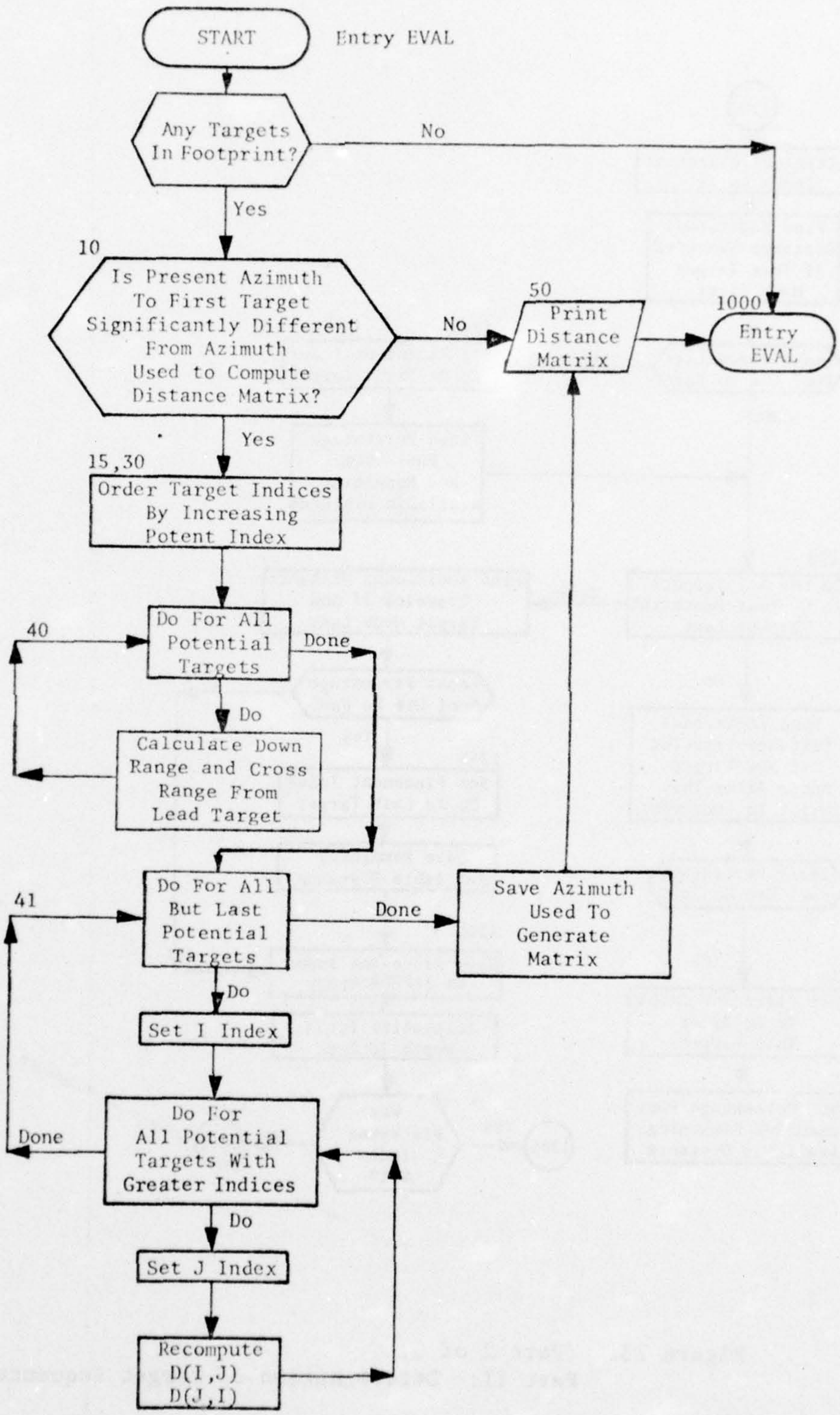


Figure 15. (Part 3 of 3)
Part III: Entry REVAL

2.6.9 Subroutine FOOTEST

PURPOSE: This routine tests footprint feasibility.

ENTRY POINTS: FOOTEST

FORMAL PARAMETERS: None

COMMON BLOCKS: C1, CSYS4, EARTH, FOOTDATA, FOOTIO, PARAMTR, PENADD, POTENT, RANGE, SHRTDAT

SUBROUTINES CALLED: ABORT, CRSTODWN, UPTODOWN

CALLED BY: ADDRV, TEST

Method:

Most of the input/output for this subroutine is contained in common /FOOTIO/. Data on the target set to be tested are contained in the arrays RIN and THIN (for range and launch azimuth, respectively). Subroutine FOOTEST computes the equivalent downrange distance between each successive target in the footprint. It then determines the number of RVs that can be delivered to the target set without violating the fuel consumption constraints. If an RV can be delivered to each target in the set, then this subroutine computes the effect of using the total remaining fuel load to deliver one more RV. It outputs the maximum equivalent downrange distance that the remaining fuel will allow from each point in the footprint.

Figure 17 displays the processing flow for this routine. Since the methods used to test footprints are essentially the same for all systems, only the long-range method will be described. The other system methods differ only in the details of processing.

Part I: Distance Computation

Before testing the footprint for feasibility, the routine calls functions CRSTODWN and UPTODOWN to retrieve the correct downrange-crossrange and downrange-uprange distance between successive targets in the footprint. These distances are placed in array DT, which is equivalenced to array TOFLY in common/POTENT/. This subroutine does not use function FLYDIST for the distance computation, but rather computes the equivalent distance from the basic

range and azimuth data. There are two reasons for this independent calculation. First, this subroutine can test any data contained in common /FOOTIO/, without considering the data in the potential target arrays. Second, the footprint testing subprograms, FOOTEST, CRSTODWN, UPTODOWN, SETDATA, and TABLINPT, which comprise the testing module, were designed to be as separate as possible from the other subprograms. This modular design allows for modification of either the assignment module subprograms or the test module subprograms without excessive manipulation of the interface between the modules.

Part II-A: Long-Range System

In the following discussion, a "leg" of a footprint refers to the line between two successive targets in the footprint. The Jth leg will be the line between target J and target J+1. The length of these legs (in terms of equivalent downrange distance) determines the feasibility of the footprint.

The testing algorithm (for the long-range system) begins with a determination of the number of reentry vehicles in the footprint. A number indicator JRV (or NRV) is set to specify the correct set of footprint parameter constraint equations to be used. These equations will vary according to the number of RVs on board the bus. In order to save processing time, FOOTEST precomputes all the necessary fuel consumption and booster range parameters and stores them in a set of temporary arrays (e.g., RSAVE, REXSAVE, and CSAVE). These parameters will change only if there is a new first target in the footprint with a significantly different range (or azimuth for the non-short range system). Thus, on each call to FOOTEST, the range and azimuth of the first target are tested against the previous values for these factors. If either factor differs from the saved value by an amount greater than or equal to PDIFF (a preset test variable), then the range and fuel parameters are recomputed. The larger the preset value of PDIFF, the fewer times these parameters will be recomputed.

If the long range system has a full load of penetration aids (i.e., MTYPE = 3), then the fuel load at booster separation is computed by a special set of equations (statement 3000).

The main testing algorithm begins at statement 1308 with a calculation of the total fuel load available for footprinting and the maximum booster range. If the range to the first target exceeds the maximum booster range, some of the fuel is used for range extension. This range extension fuel is subtracted from the total load available for footprinting. If this subtraction results in a negative fuel load then the subroutine returns with the feasibility indicator, IFEAS, set to 0.

This variable contains the number of targets in the footprint that can be reached within fuel constraints.

Part II-B: Long-Range System (continued)

If there is some fuel left for use on the legs, the feasibility indicator is set to 1 and the number of reentry vehicles to be delivered NTOGO is set to the original number minus one (since one RV has been delivered to the first target point). FOOTEST then computes the fuel use on each leg. It retrieves the correct fuel consumption rate for the current load and the equivalent downrange distance for the leg. A division gives the amount of fuel used on the leg. If there is not sufficient fuel left for that leg, the fuel remaining indicator FUELEFT is set to 0 (statement 1365), and the routine returns control to the calling program. If there is sufficient fuel for the leg, the fuel remaining is decremented by the fuel used on the leg, the feasibility indicator is incremented, and the number of RVs on board is decremented. Then the next leg is tested.

When all the legs have been tested the fuel left after footprinting FUELEFT is saved (statement 1390). If the booster is currently carrying the maximum allowed load, control returns to the calling program. If more reentry vehicles can be added, the best use of the extra fuel is calculated (starting at statement 1396).

Part II-C: Long-Range System (continued)

This section begins by resetting the initial load indicator to show the potential addition of another reentry vehicle to the original payload. (If necessary, the number indicator, JRV or NRV, is reset.) The same computations as were done previously to test the footprint are repeated with the increased load. This time, however, the difference in fuel use between the original load and the increased load is saved in array EXTRA. The value of the element EXTRA(J) is the amount of extra fuel that would be needed on leg J (from target J to target J+1) to carry one more RV. These computations are performed in the "do loop" ending on statement 1420.

Then, this extra required fuel is subtracted, cumulatively, from the fuel left after completion of the footprint ("do loop" ending on statement 1430). The elements of the array EXTRA are changed to contain the successive results of these subtractions. The contents of EXTRA(J) now contain the fuel that would be available for further footprinting if a new target were added to the footprint between target J and target J+1. The algorithm assumes the extra reentry vehicle is carried on the bus for the deliveries through target J. Then the extra reentry vehicle is delivered to another target and the bus proceeds as before. The amount

of fuel that could be used for the extra flying distance created by insertion of a new target is now contained in the EXTRA array.

This extra fuel available for further footprinting is now converted into a maximum allowable distance. The testing algorithm assumes that all the extra fuel would be used by the bus to deliver the added reentry vehicle. (Note that the fuel needed to complete the footprint after that delivery is reserved and cannot be used for the addition.) Thus, the extra fuel for each leg is multiplied by the saved consumption rate for each leg, CR, to calculate the maximum extra flying distance allowed on the leg. This distance is stored in array DELRAFT in common /FOOTIO/. Subroutine EVAL uses this distance to determine the worth of adding a new target to the footprint. Since other subprograms divide by these distances, the values placed in the array are increased to a minimum nonzero value (EPSILON, preset to .00000001) to allow that division. This completes footprint testing and control returns to the calling program.

Part III: Short-Range System

As previously stated, the methods for testing footprints for the short-range system is essentially the same as the long-range system, differing only in the details of processing. The processing flow for this function is shown in part III of figure 17.

Part IV: Type-4 System

The type-4 system is very similar to the short-range system, even to the point of sharing some flags and code. The processing flow for this function is shown in Part IV of figure 17.

Part V: Error and Termination Blocks

Part V of figure 17 is the logic flow for printing out the errors which may be encountered within the subroutine.

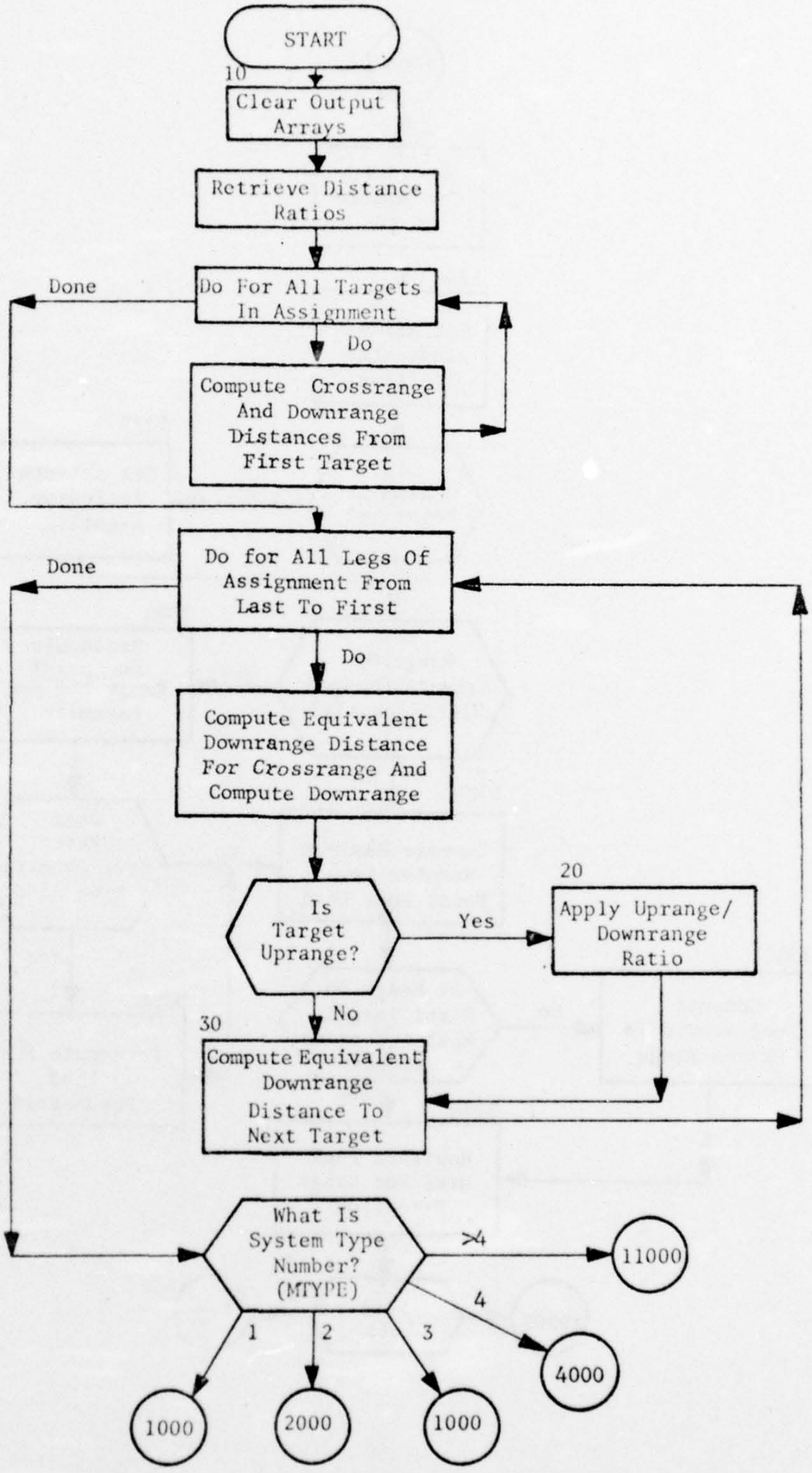


Figure 17. Subroutine FOOTEST (Part 1 of 8)
Part I: Distance Computation

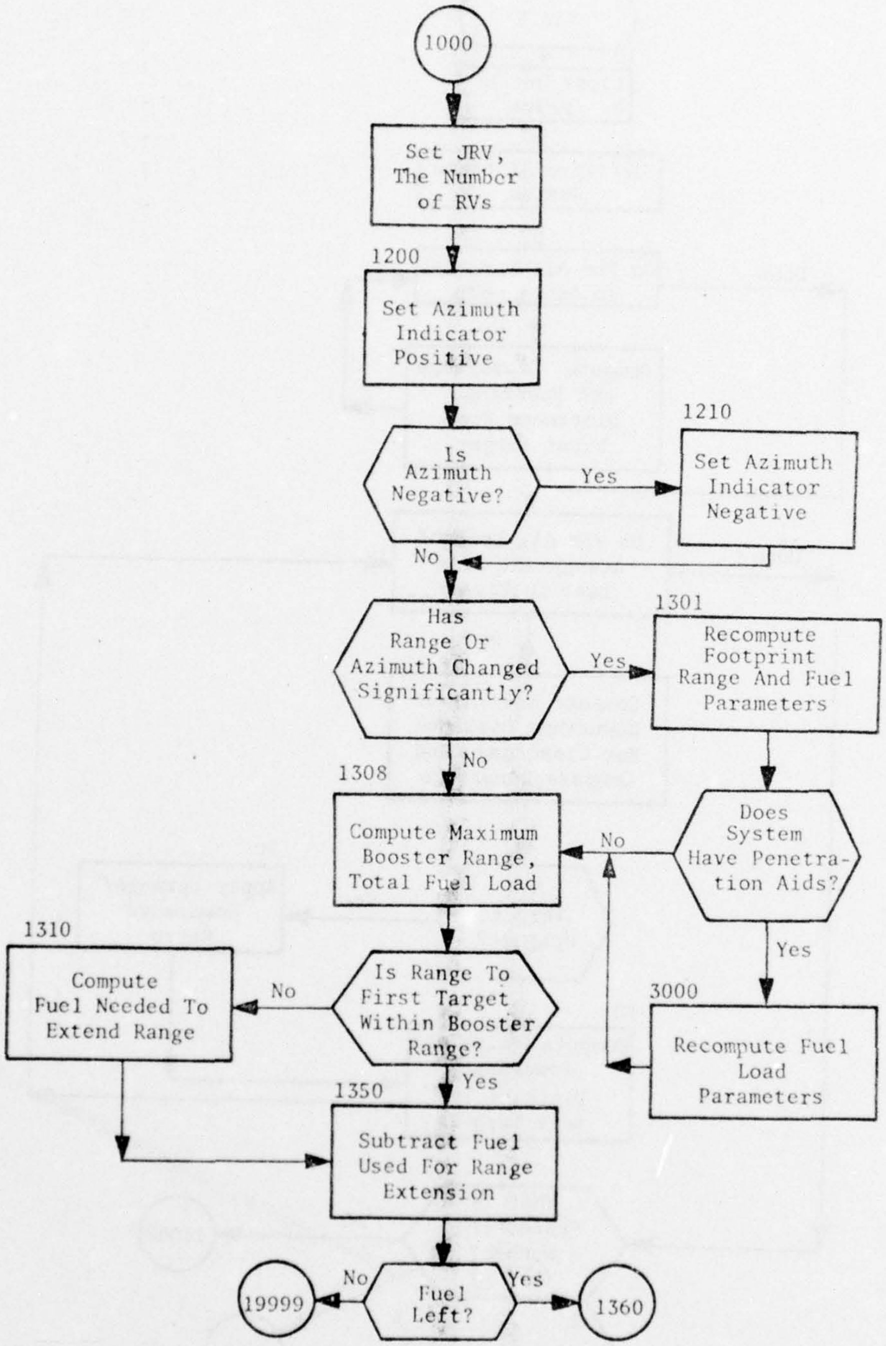


Figure 17. (Part 2 of 8)
Part II-A: Long-Range System

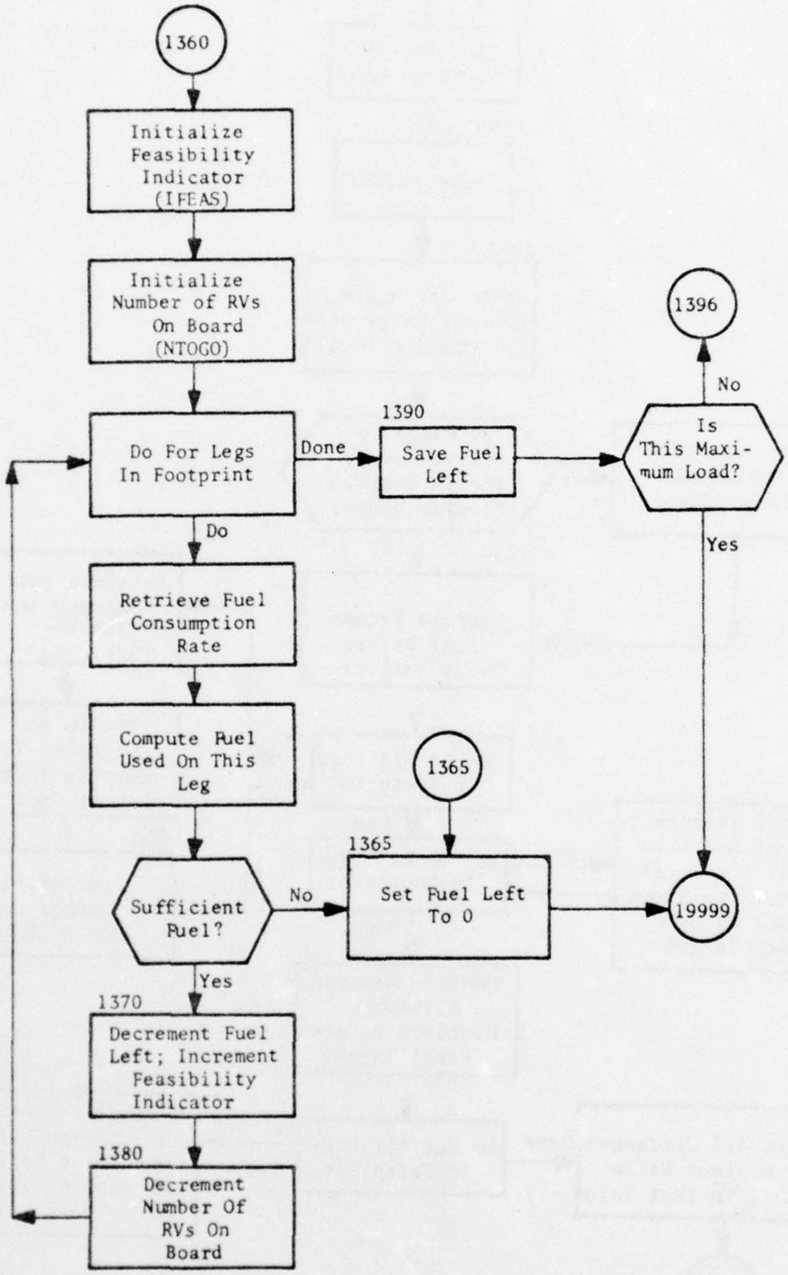


Figure 17. (Part 3 of 8)
Part II-B

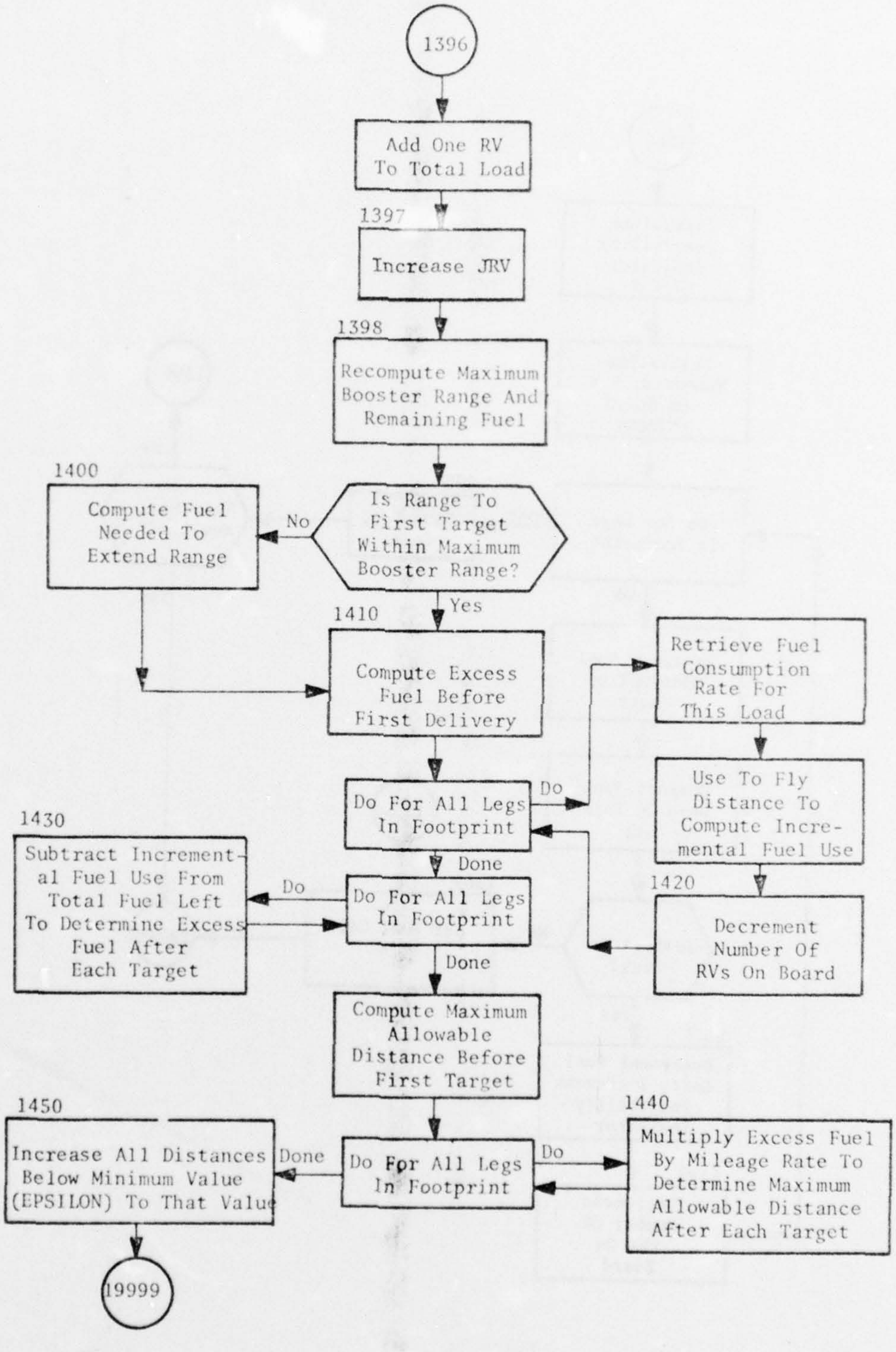


Figure 17. (Part 4 of 8)
Part II-C

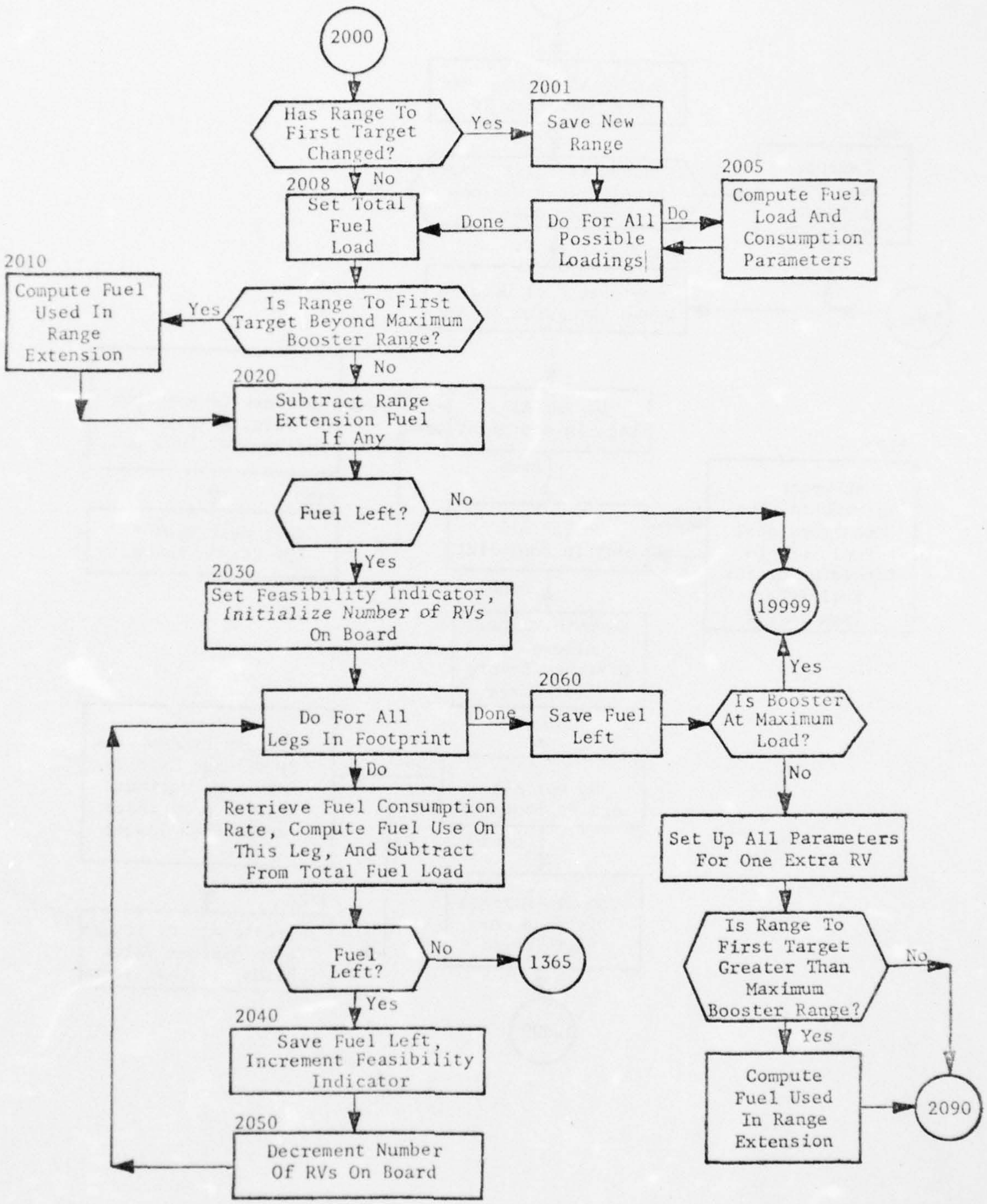


Figure 17. (Part 5 of 8)
Part III: Short-Range System

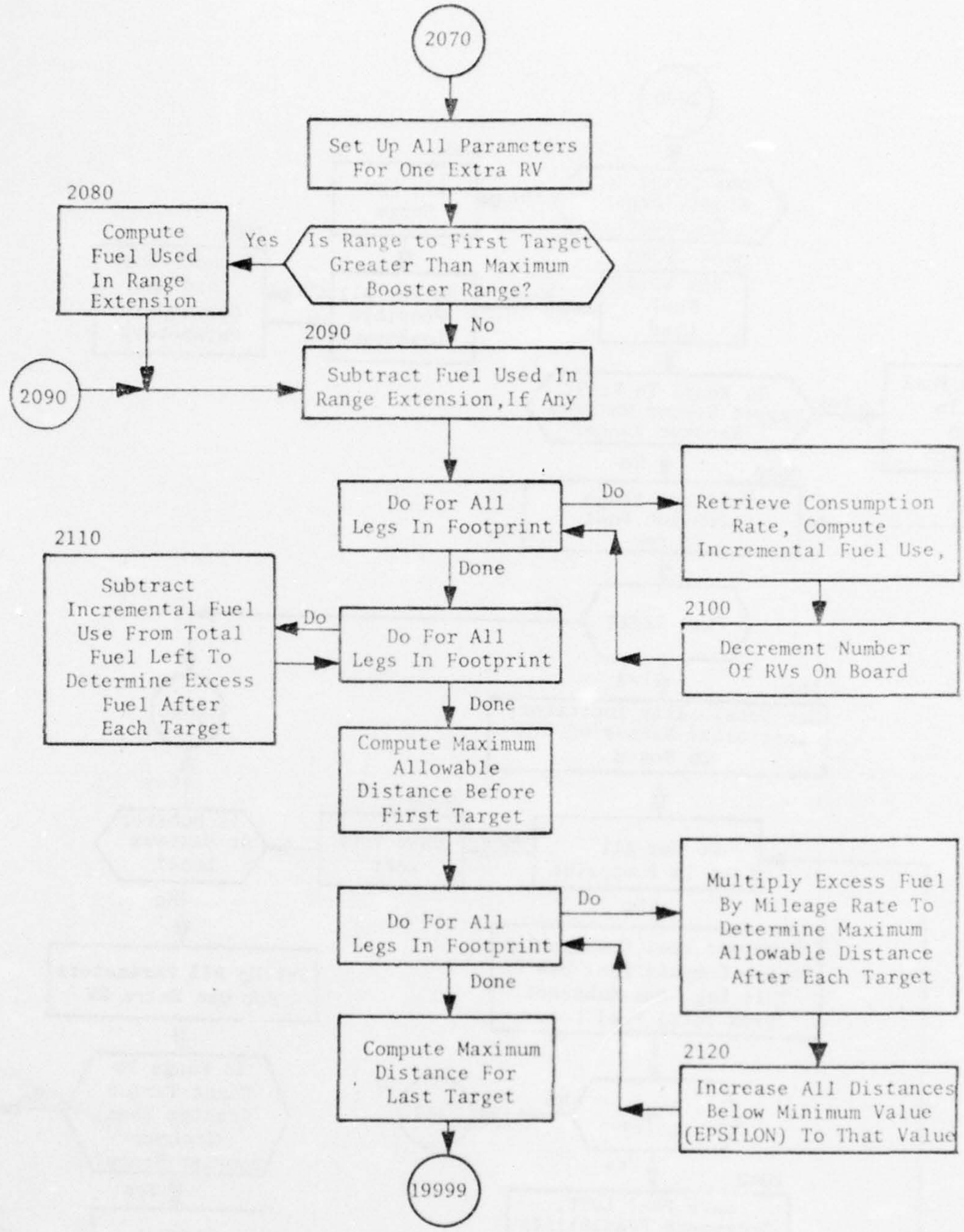


Figure 17. (Part 6 of 8)
Part III

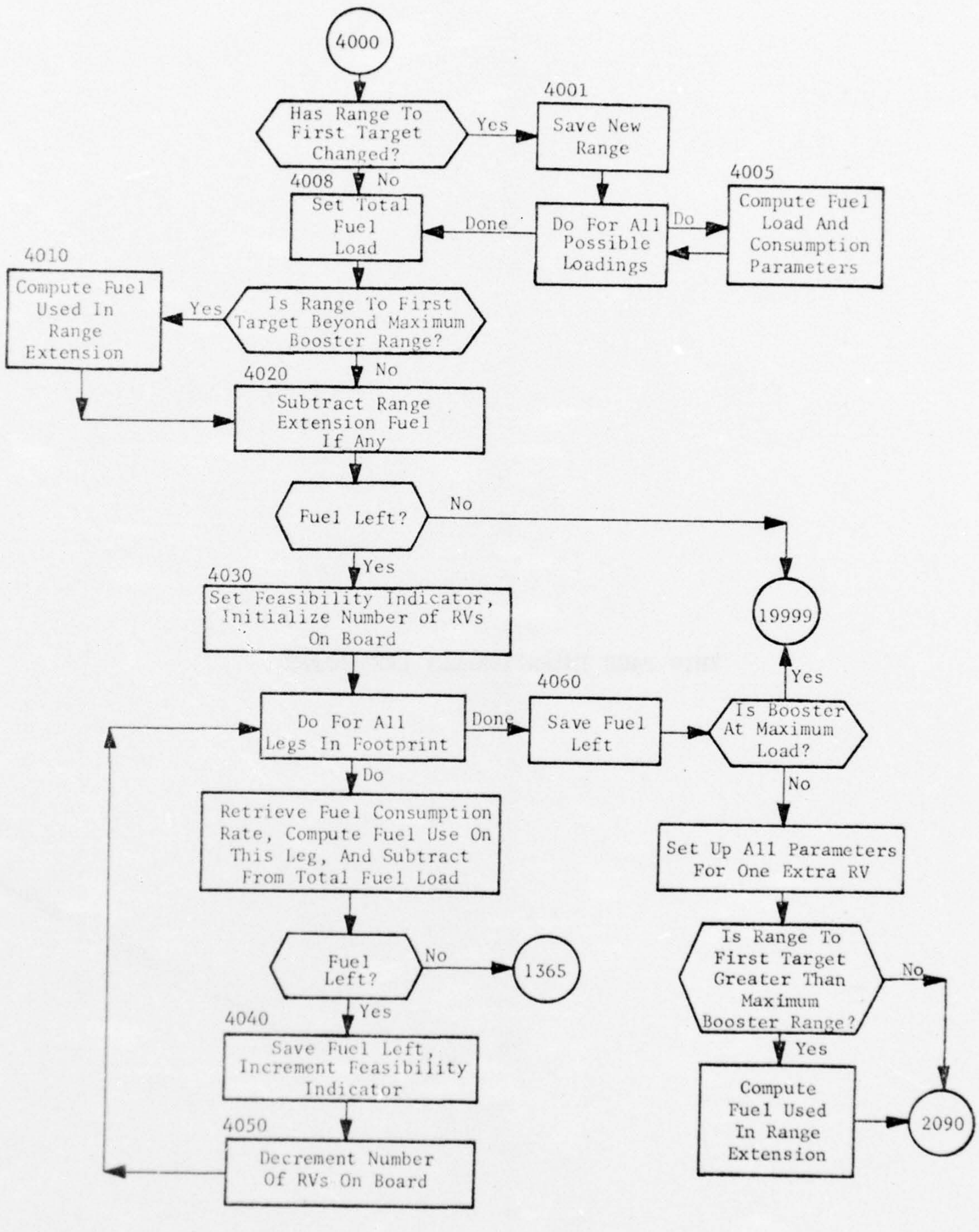


Figure 17. (Part 7 of 8)
Part IV: TYPE-4 System

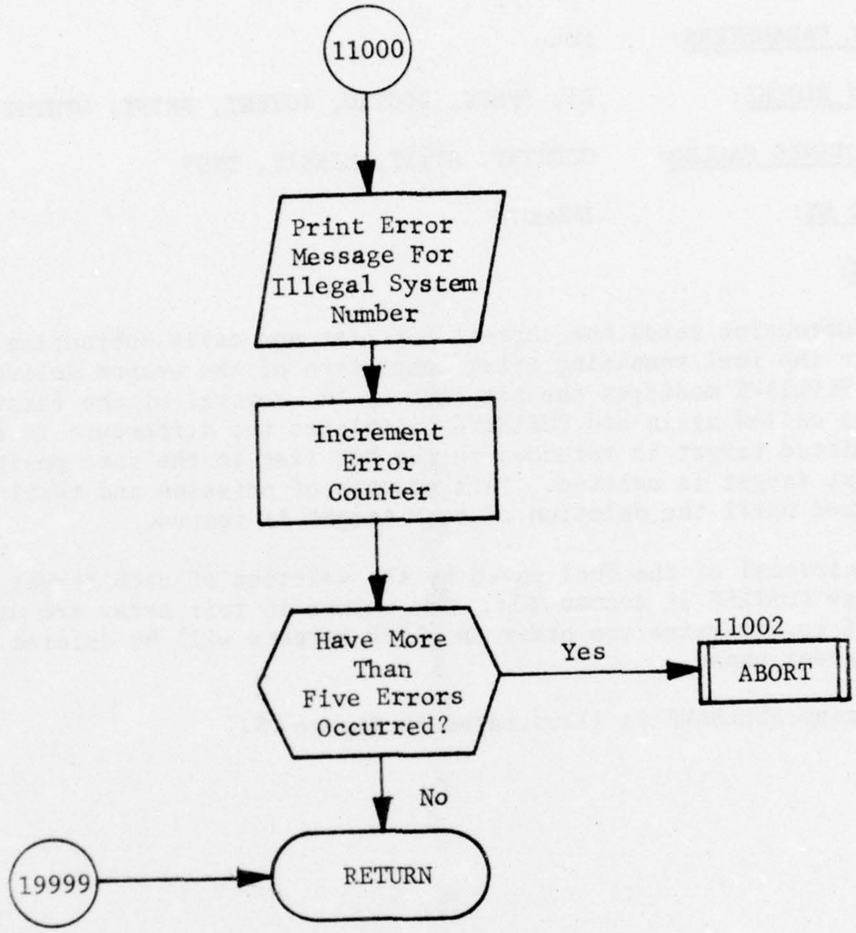


Figure 17. (Part 8 of 8)
Part V: Error and Termination Blocks

2.6.10 Subroutine FUELSAVE

PURPOSE:	This routine computes the fuel saved by the omission of each target in the footprint.
ENTRY POINTS:	FUELSAVE
FORMAL PARAMETERS:	None
COMMON BLOCKS:	C1, DEBUG, FOOTIO, POTENT, PRINT, WPNTGT
SUBROUTINES CALLED:	GOPRINT, HITIT, MISSIT, TEST
CALLED BY:	IMPROVE

Method:

This subroutine takes the current hit list and calls subroutine TEST to compute the fuel remaining after completion of the weapon deliveries. Then, FUELSAVE modifies the hit list by the removal of the first target. TEST is called again and FUELSAVE calculates the difference in fuel used. The omitted target is returned to the hit list in the same position and the next target is omitted. This process of omission and testing is continued until the deletion of each target is tested.

The reciprocal of the fuel saved by the deletion of each target is stored in array COSTEFF in common /C1/. The values in this array are used by IMPROVE to determine the order in which targets will be deleted during the improvement phase.

Subroutine FUELSAVE is illustrated in figure 18.

subroutine RDCARDF. If so, subroutine ABORT is called to produce a memory dump. Otherwise control is returned to the calling program.

Subroutine GOPRINT is illustrated in figure 19.

Table 6. Data Blocks Used in Print Requests (Part 1 of 2)

NUMBER	COMMON	DESCRIPTION
1	WPNGRPX	Group data read from BASFILE
2	PARAMETR	MIRV system general parameters
3	------	Detailed footprint parameter tables
4	------	Detailed booster loading tables (not used)
5	STRKSUM	Gross strike data block
6	RAIDATA C4 C5	Detailed strike data block
7	------	Range and launch azimuth of target set (includes index according to azimuth)
8	C2	Status array, pointer arrays, booster loadings and pointers
9	RANGE	Downrange/uprange, downrange/crossrange ratios
10	DEBUG	List of chain of subroutine calls
11	POTENT C1	Potential target arrays; includes hit, miss, lost and free lists as well as age and value arrays
12	CONTROL	Control parameters for program
13	FOOTIO	Input/output data for footprint tester
14	PERFORM	Gross performance parameters
15	------	Same as number 11, /POTENT/ and /C1/
16	WPNTGT	Indices for moving targets between hit and miss lists (includes target to booster assignment indices)
17	C3 RANGE	Temporary storage of various parameters for all targets in potential target arrays. (includes common /RANGE/, number 9)

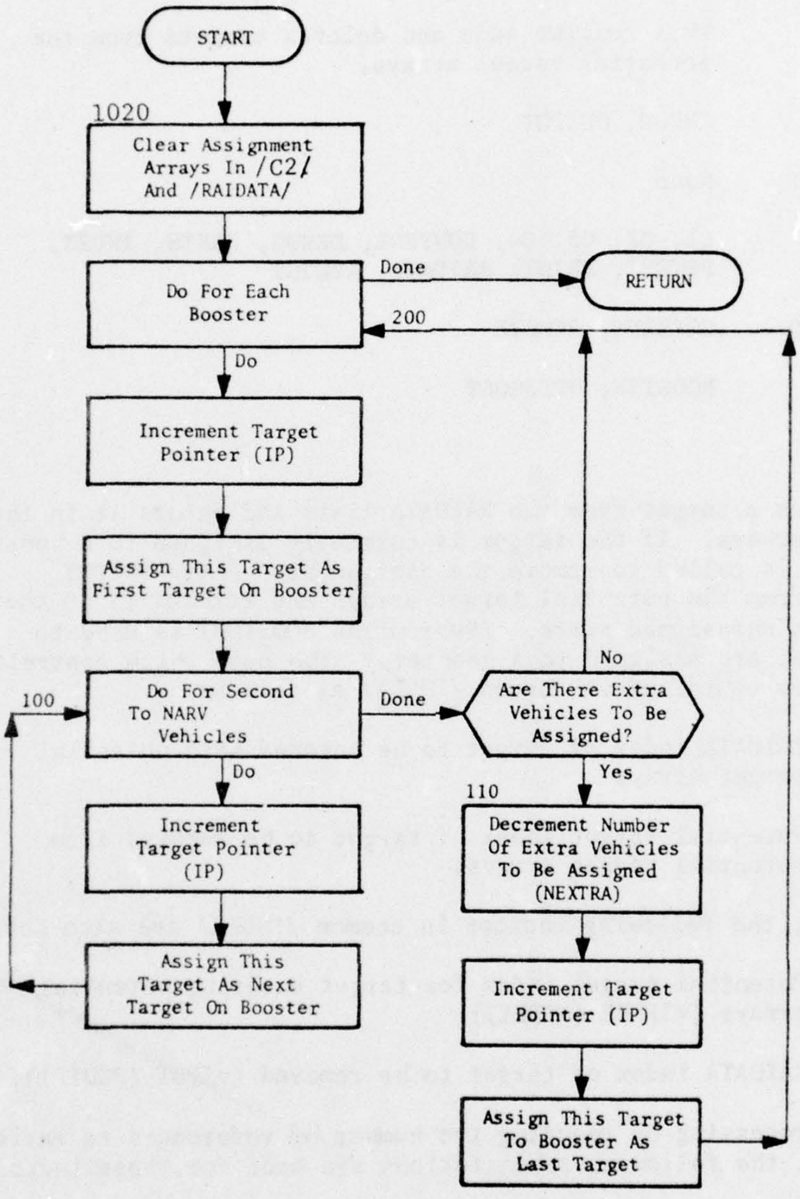


Figure 22. Subroutine INITASGN

2.6.15 Subroutine INPOT

PURPOSE: This routine adds and deletes targets from the potential target arrays.

ENTRY POINTS: INPOT, OUTPOT

FORMAL PARAMETERS: None

COMMON BLOCKS: C1, C2, C3, C4, CONTROL, DEBUG, EARTH, INDEX, POTENT, PRINT, RAIDATA, WPNTGT

SUBROUTINES CALLED: GOPRINT, REMOVE

CALLED BY: BOOSTIN, OPTBOOST

Method:

Entry INPOT removes a target from the RAIDATA lists and enters it in the potential target arrays. If the target is currently assigned to a booster, subroutine REMOVE is called to remove the assignment. Entry OUTPOT removes a target from the potential target arrays and returns it to the RAIDATA list in an unassigned state. (Subroutine BOOSTOUT is used to remove targets that are assigned to a booster.) The data which controls this subroutine are contained in common /INDEX/ as follows:

JINR - RAIDATA index of target to be entered into potential target arrays

JOUTP - Potential target index of target to be removed from potential target arrays.

During processing, the following indices in common /INDEX/ are also defined:

JINP - Potential target index for target entering potential arrays (=IFREE (NFREE))

JOUTR - RAIDATA index of target to be removed (=IPOT (JOUTP)).

To save time in processing by reducing the number of references to variables in common storage, the following substitutions are made for these indices:

JR = JINR (in INPOT)

JR = JOUTR (in OUTPOT)

JP = JINP (in INPOT)

JP = JOUTP (in OUTPOT)

The processing of this subroutine is very straightforward, as displayed in figure 23.

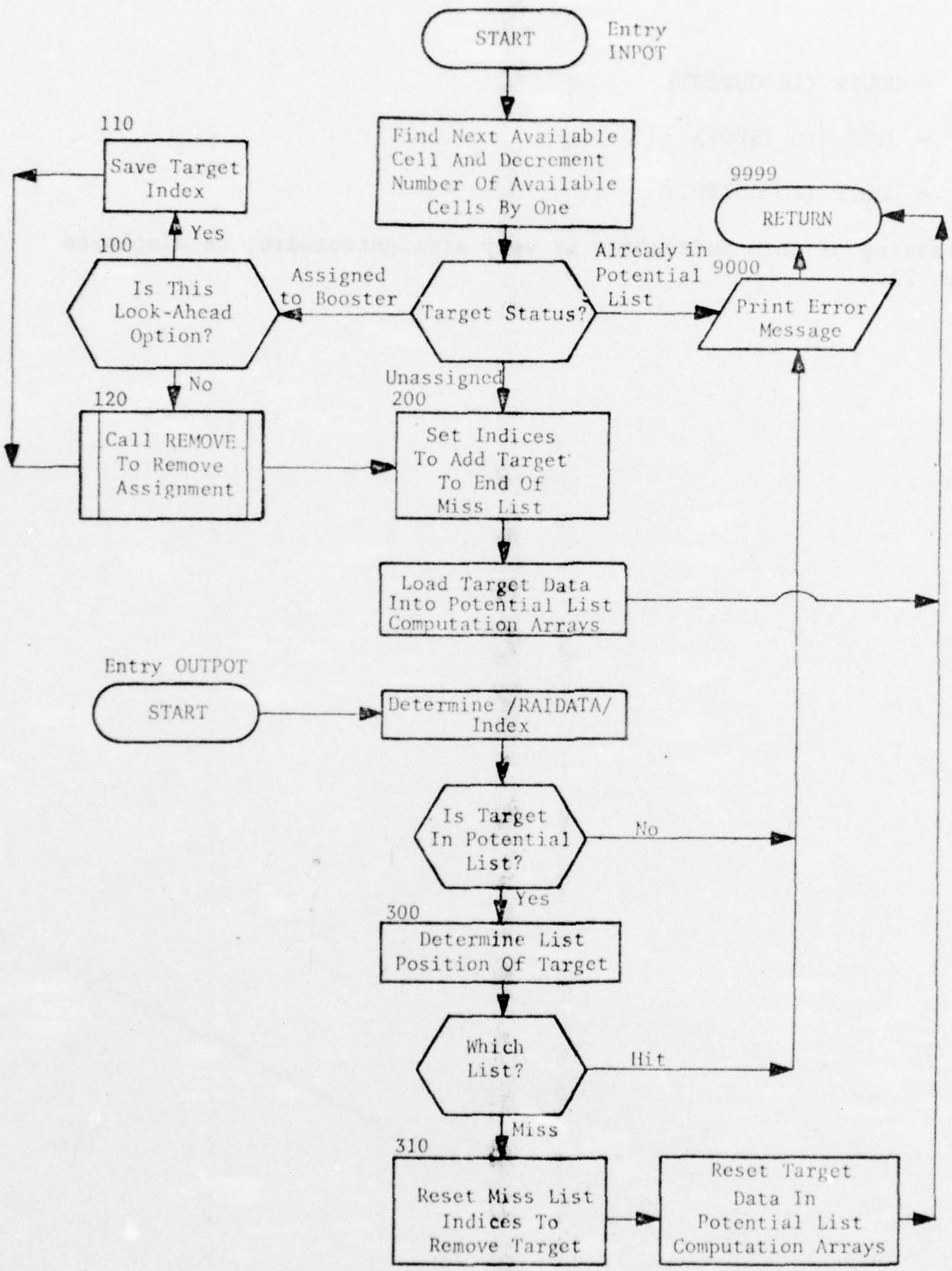


Figure 23. Subroutine INPOT

2.6.16 Subroutine LOADREAD

PURPOSE: This routine reads and prints booster loading data.

ENTRY POINTS: LOADREAD, PRNTLOAD

FORMAL PARAMETERS: None

COMMON BLOCKS: None

SUBROUTINES CALLED: None

CALLED BY: GOPRINT (PRNTLOAD), RDCARDF (LOADREAD)

Method:

This subroutine is currently a dummy routine. Its purpose will be to read data on variable booster loadings within a group and also to print that data. The dummy routine merely reserves the entry points for later expansion of the program to include a variable booster loading option. (LOADOPT = *VARY* = option 2)

2.6.17 Subroutine NEWCOOR

PURPOSE	This routine converts target coordinates from latitude and longitude to range and azimuth from weapon group centroid.
ENTRY POINTS:	NEWCOOR
FORMAL PARAMETERS:	IG - Group Number
COMMON BLOCKS:	C2, C4, DEBUG, EARTH, FILES, Filehandler Blocks (ITP, MYIDENT, TWORD, NOPRINT, FILABEL), HDATA, IFTPRNT, MYLABEL, PRINT, RAIDATA, TSCRATCH, WPNGRPX
SUBROUTINES CALLED:	DISTF, GOPRINT, LREORDER, ORDER, REORDER, WRARRAY
CALLED BY:	FOOTPRNT

Method:

This routine converts the target coordinates for use by the footprint generation subroutines. It is called once for each group.

For each target, NEWCOOR adds the target point offsets (DLAT, DLONG) to the target coordinates (TGTLAT, TGTLONG). The range from the weapon group centroid to the target point (RANGE) is then computed by a call on the distance function, DISTF. The position of the group centroid is given by the variables WLAT and WLONG in common /WPNGRPX/. The formal parameter IG is used to retrieve the correct position.

The calculation of the launch azimuth uses spherical trigonometry. First all the latitudes are converted to radians by the factor DEGTORAD in common /EARTH/. The range is normalized by dividing by the radius of the earth (RADIUS in common /EARTH/).

The computation of the launch range is performed as follows. Define a spherical triangle with vertices at the group centroid, North Pole and target. (See figure 24.) Let angle A (the launch azimuth) be the angle between the line connecting the centroid and the North Pole and the line connecting the centroid and the target. Measure the distances between the points in terms of the number of radians subtended by the connecting lines. If distance a is the distance between target and North Pole, b is the distance between centroid and target, and c is the distance between centroid and North Pole, then:

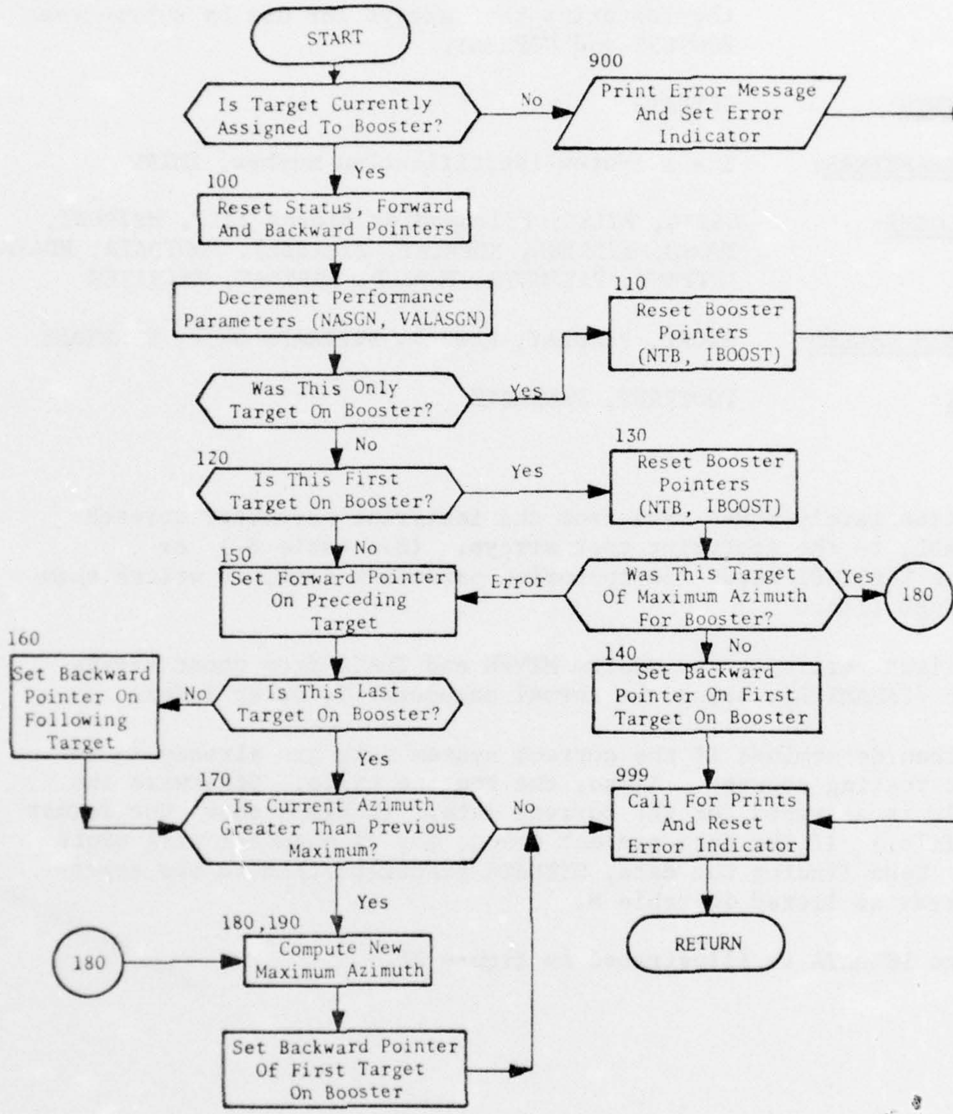


Figure 30. Subroutine REMOVE

2.6.23 Subroutine SETDATA

PURPOSE: This routine retrieves (from the ITABL file) and loads the correct footprint processing data into the footprint test arrays for use by subroutines FOOTEST and GOPRINT.

ENTRY POINTS: SETDATA

FORMAL PARAMETERS: I - A system identification number, IMIRV

COMMON BLOCKS: CSYS4, FILES, Filehandler Blocks (ITP, MYIDENT, TWORD, MYLABEL, NOPRINT, FILABEL), FOOTDATA, HDATA, IFTPRNT, PARAMETR, PENADD, SHRTDAT, TSCRATCH

SUBROUTINES CALLED: ABORT, RDARRAY, RDWORD, SETREAD, SKIP, TERMTAPE

CALLED BY: FOOTPRNT, PRNTABLE

Method:

This routine merely moves data from the footprint parameter scratch file, ITABL, to the footprint test arrays. (See table 8.) As subroutine TABLINPT reads the footprint parameter data, it writes them on the ITABL file.

SETDATA first retrieves the system MTYPE and IDATA from those arrays in common /PARAMETR/, using the formal parameter I, as an index.

SETDATA then determines if the correct system data are already in the footprint testing storage. If so, the routine exits. Otherwise the ITABL file is searched for the correct data. (Table 9 shows the format of this file.) If the data are not found, the filehandler will abort the run. Upon finding the data, SETDATA transfers them to the appropriate array as listed in table 8.

Subroutine SETDATA is illustrated in figure 31.

Table 8. Footprint Parameter Data Transmission

MTYPE	SYSTEM	ARRAY LENGTH	FOOTPRINT TESTING COMMON BLOCK
1	Long-Range	LLNGDAT	/FOOTDATA/
2	Short-Range	LSHTDAT	/SHRTDAT/
3	Long-Range With Penetration Aids	LPENDAT	/PENADD/ /FOOTDATA/
4	Type-4	LDSYS4	/CSYS4/

Table 9. Format for Footprint Parameter Data Scratch File

Each unique system is output on the ITABL file in the following format:

VARIABLE	LENGTH	DESCRIPTION
MTYPE	1	MIRV system functional type
IDATA	1	MIRV system data set number
"LENGTH"*	1	Length of footprint parameter table for this system
"TABLE"**	LENGTH	Footprint parameter table

*For MTYPE=1, LENGTH is LLNGDAT; MTYPE=2, LENGTH is LSHTDAT; MTYPE=3, LENGTH is LPENDAT; MTYPE=4, LENGTH is LDSYS4 (see table 8).

**For MTYPE=1, TABLE is the FOOTDATA common; MTYPE=2, TABLE is the SHRTDAT common; MTYPE=3, TABLE is the PENADD and FOOTDATA commons; MTYPE=4, TABLE is the CSYS4 common (see table 8).

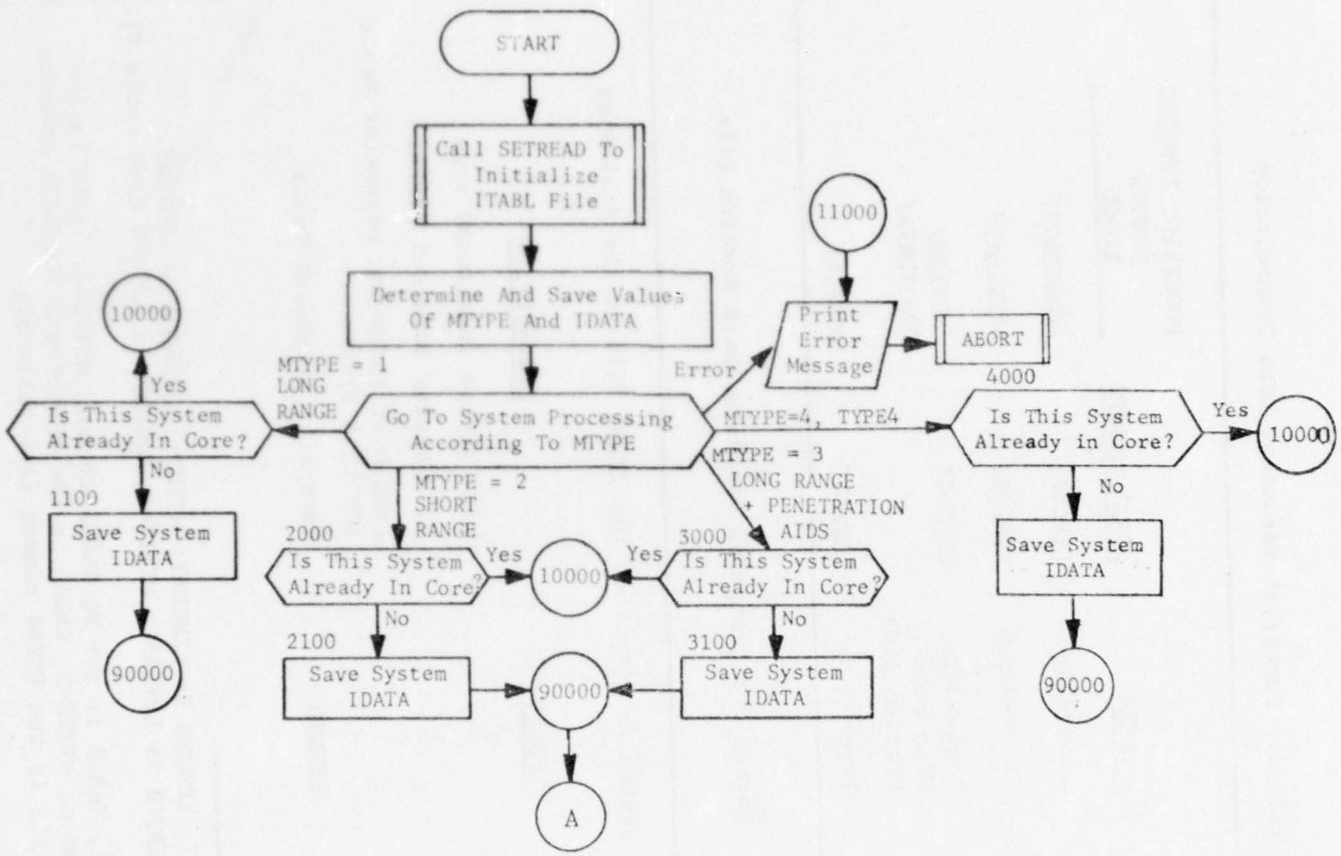


Figure 31. Subroutine SETDATA
(Part 1 of 2)

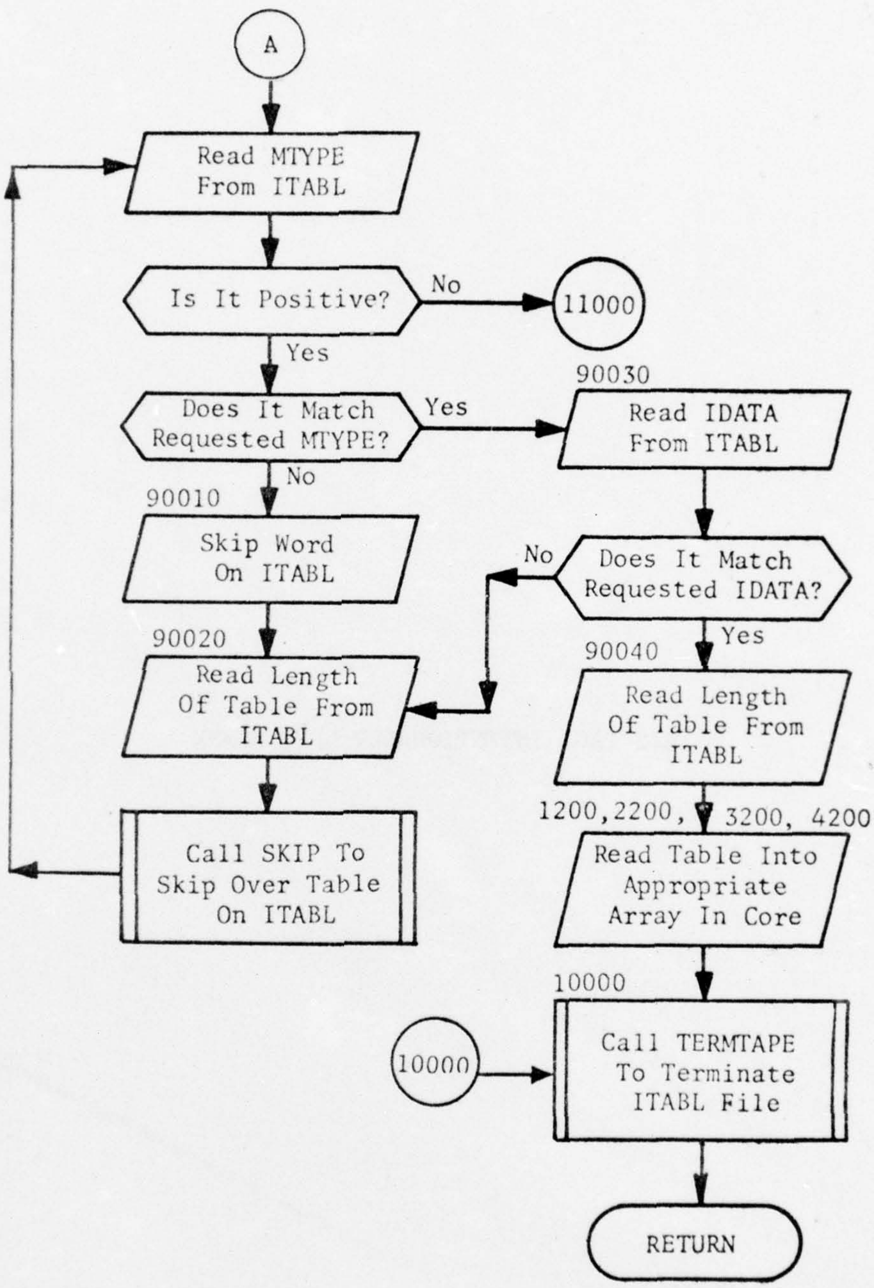


Figure 31. (Part 2 of 2)

2.6.24 Subroutine TABLINPT

PURPOSE:	This routine reads and prints the footprint parameter tables, and saves them on the ITABL file.
ENTRY POINTS:	TABLINPT, PRNTABLE
FORMAL PARAMETERS:	None
COMMON BLOCKS:	C4, CSYS4, FILES, Filehandler Blocks (ITP, MYIDENT, MYLABEL, TWORD, NOPRINT, FILABEL), FOOTDATA, HDATA, IFTPRNT, PARAMETR, PENADD, RAIDATA, SHRTDAT, TSCRATCH, WAROUT
SUBROUTINES CALLED:	NUMGET, SETDATA, SETWRITE, TERMTAPE, WRARRAY, WRWORD
CALLED BY:	RDCARDF (TABLINPT), GOPRINT (PRNTABLE)

Method:

This routine reads the footprint parameter tables and stores the data in common block /PARAMETR/ and outputs the data on the footprint parameter data scratch file, ITABL. Entry PRNTABLE calls subroutine SETDATA to transfer the data from the ITABL file to the footprint testing parameter arrays (in /FOOTDATA/, /SHRTDAT/, /PENADD/, and /CSYS4/) and then prints the data. Common /RAIDATA/ is used as temporary scratch storage by this subroutine. The local array IGOT is used to store the IMIRV numbers of those systems whose parameters have been read. NGOT is the number of system data sets that have been read.

Each MIRV system with a unique IMIRV number must be defined with a system title card.* The data on this card are stored in common /PARAMETR/. (See table 3 for definitions of the variables in this block.)

The data required for each data set depends on the system type number, MTYPE. This variable defines the system to be long-range, short-range, long-range with penetration aids or Type-4. (See table 3.) Within each type, there may be many different data sets identified by the data set number, IDATA. (The values of this parameter need be unique only within each MTYPE value. The values need not be consecutive.) As each data set is read, it is output on the ITABL file according to the format shown in table 9. A data set need be read only once regardless of the number of systems that use it. If the values of MTYPE and IDATA read from a system title card match values already read, then the routine merely reads the next title card.

---

*For each value of the attribute IMIRV, there should be one title card. The Hollerith name of the system (IHNAME in common /PARAMETR/) is used only to identify the system in the print of the footprint parameter tables. It has no effect on footprint generation.

Entry PRNTABLE retrieves the data for each defined system and prints the data.

Each formula's data cards are preceded by one system title card requesting that formula. The reading of data is terminated by a title card with a zero or negative IMIRV value. The systems can be input in any order.

If more than one IMIRV value refers to a specific formula for footprint test (see below), then the data for that formula must follow immediately the first occurrence of a system title card requesting the use of that formula. Succeeding title cards with the same formula definition need no data following them.

A formula for footprint testing is defined by two variables input on the system title card. The first, MTYPE, references the functional form of the formula to be used. If MTYPE = 1, the exponential functions of the long-range system are used. MTYPE = 2 requests the short-range functions. MTYPE = 3 requests the long-range system with a full load of area penetration aids. MTYPE=4 requests the type-4 functions. Within each type, there are data sets for the parameters used in the function. Thus, formula definition requires MTYPE, the functional form indicator, and IDATA, the index to the parameter set. For example, if two long-range systems are desired there would be two formula definitions: MTYPE=1, IDATA=1; MTYPE=1, IDATA=2.

Long-Range System -- MTYPE=1

The long-range system can have one to 16 reentry vehicles on a booster.

The system functions are defined by a series of regression coefficients which, when applied to these functions, produce results which fit the actual physical characteristics of the MIRV system.

The system functions are as follows:

a. Fuel Load at Booster Separation (Pounds): Constant with number of RVs.

b. Maximum Booster Range (RM in Nautical Miles):

RM = RBASIC + RADD * SINE(AZIMUTH)

RBASIC and RADD are functions of the number of RVs and the sign of the azimuth.

c. Range Extension Consumption: number of nautical miles traversed per pound of fuel

NM/FUEL = RX + RAXX * SINE(AZIMUTH)

RX and RAXX are functions of the number of RVs and the sign of the azimuth.

d. RV Toss Equations: nautical miles per unit fuel

NM/FUEL = G * (TOSSC1 + TOSSC2 * SINE(AZIMUTH))

where

$$G = \mathrm{EXPF}\left(\mathrm{TEONE} * \left(\frac{\mathrm{RM-R}}{\mathrm{TDENOM}}\right)^{**\mathrm{TETWO}}\right)$$

where

RM = maximum booster range (nautical miles)
R = range to initial target (nautical miles)

TOSSC1 and TOSSC2 are functions of number of RVs originally on board, number of RVs currently on board, and sign of launch azimuth.

TEONE and TETWO are functions of number of RVs originally on board and number currently on board.

TDENOM is a constant.

e. Downrange to Crossrange Multiplier (CROSSDWN):

CROSSDWN = G * (CONE + CTWO * SINE(AZIMUTH))

where

$$G = \mathrm{EXPF}\left(\mathrm{EONE} * \left(\frac{\mathrm{RM-R}}{\mathrm{DENOM}}\right)^{**\mathrm{ETWO}}\right)$$

CONE and CTWO are functions of the number of RVs currently on board and the sign of the azimuth.

EONE and ETWO are functions of the number of RVs.

DENOM is a constant.

f. <u>Downrange to Uprange Multiplier (UPTODOWN)</u>

UPTODOWN = G * (UC1 + UC2 * SINE(AZIMUTH))

where

$$G = \text{EXPF}\left(\text{UE1} * \left(\frac{\text{RM-R}}{\text{UDEN}}\right)^{**\text{UE2}}\right)$$

UC1 and UC2 are functions of the number of RVs currently on board and the sign of the azimuth.

UE1 and UE2 are functions of the number of RVs.

UDEN is a constant.

Short-Range System -- MTYPE=2

This system does not consider launch azimuth. It considers configurations containing from 1 to 16 RVs on board. The system functions are as follows: (Let R be the distance in nautical miles from the launch base to the initial target in the footprint.) The parameters for this type are also coefficients calculated by a curve fit to observed physical data.

a. Fuel Load at Booster Separation:

$$TF = BETATWO * R^2 + BETONE * R + BETAZ$$

The parameters are functions of the number of RVs on board.

b. Maximum Booster Range: This is a parameter, MAXRBOOST, as a function of the number of RVs carried to the first target.

c. RV Toss Consumption Equations:

$$NM/unit\ fuel = ALPHATWO * R^2 + ALPHAONE * R + ALPHAZ$$

These parameters are functions of the number of RVs on board.

d. **Downrange to Crossrange Multiplier:**

$$CROSSDWN = GTWO * R^2 + GONE * R + GZERO$$

These parameters are constant.

e. **Downrange to Uprange Multiplier:**

$$UPDOWN = DONE * R + DZERO$$

These parameters are constant.

Long-Range System With Penetration Aids: MTYPE=3

This system is similar to the long-range system (MTYPE=1). The equation forms are the same except for the first set, fuel load at booster separation. All the other constraints have the same functional form as the previous type.

Calculation of the fuel load at booster separation is as follows:

a. Fuel Available for Footprinting: (FAFF in pounds)

$$FAFF = TGAS - SRF$$

TGAS - Total fuel load on board last state (pounds)
SRF - Fuel required to space and release penetration aids and reentry vehicles

b. Spacing and Release Fuel: (SRF in pounds)

SRF = G * (SRFC1 + SRFC2 * SINE(AZIMUTH))

where

$$G = EXPF\left(SRFEXP1 * \left(\frac{RM-R}{SRFDEN}\right)^{**SRFEXP2}\right)$$

where

RM = maximum booster range in nautical miles
R = range from launch base to first target in footprint.

SRFDEN is a constant.

SRFC1, SRFC2, SRFEXP1, and SRFEXP2 all depend on the number of RVs initially on board the booster.

Note: The long-range system with MTYPE=1 is a special case of this type. For the former system, the spacing and release fuel is considered to depend only on the number of RVs initially on board. Thus the detailed computation of this fuel is unnecessary.

Type-4 System -- MTYPE=4

The Type-4 system can have one to 16 RVs on board. The parameters for this type are coefficients calculated by a curve fit to observed data.

a. RV Toss Fuel Consumption Equations:

$$NM/unit\ fuel = A2 * R^2 + A1 * R + A0$$

R is the range from the launch base to the initial target. The parameters are functions of the number of RVs on board.

b. Fuel Load at Booster Separation:

$$TF = B2 * R^2 + B1 * R + B0$$

The parameters are functions of the number of RVs on board.

c. Maximum Booster Range:

RM = BRANGE + BRADD * SINE(AZIMUTH)

The parameters are functions of the number of RVs on board and the sign of the azimuth.

d. Downrange to Crossrange Multiplier:

$$CROSSDWN = C' + 1 - C'^{**(R*SINE(AZIMUTH)/CRDEN)}$$

for positive azimuths and

$$CROSSDWN = C' - 1 + C'^{**(-R*SINE(AZIMUTH)/CRDEN)}$$

for negative azimuths

where $C' = CR2 * R^2 + CR1 * R + CR0$

These parameters are constants.

e. Downrange to Uprange Multiplier:

$$UPDOWN = UD2 * R^2 + UD1 * R + UD0$$

The parameters are constants.

Subroutine TABLINPT is illustrated in figure 32.

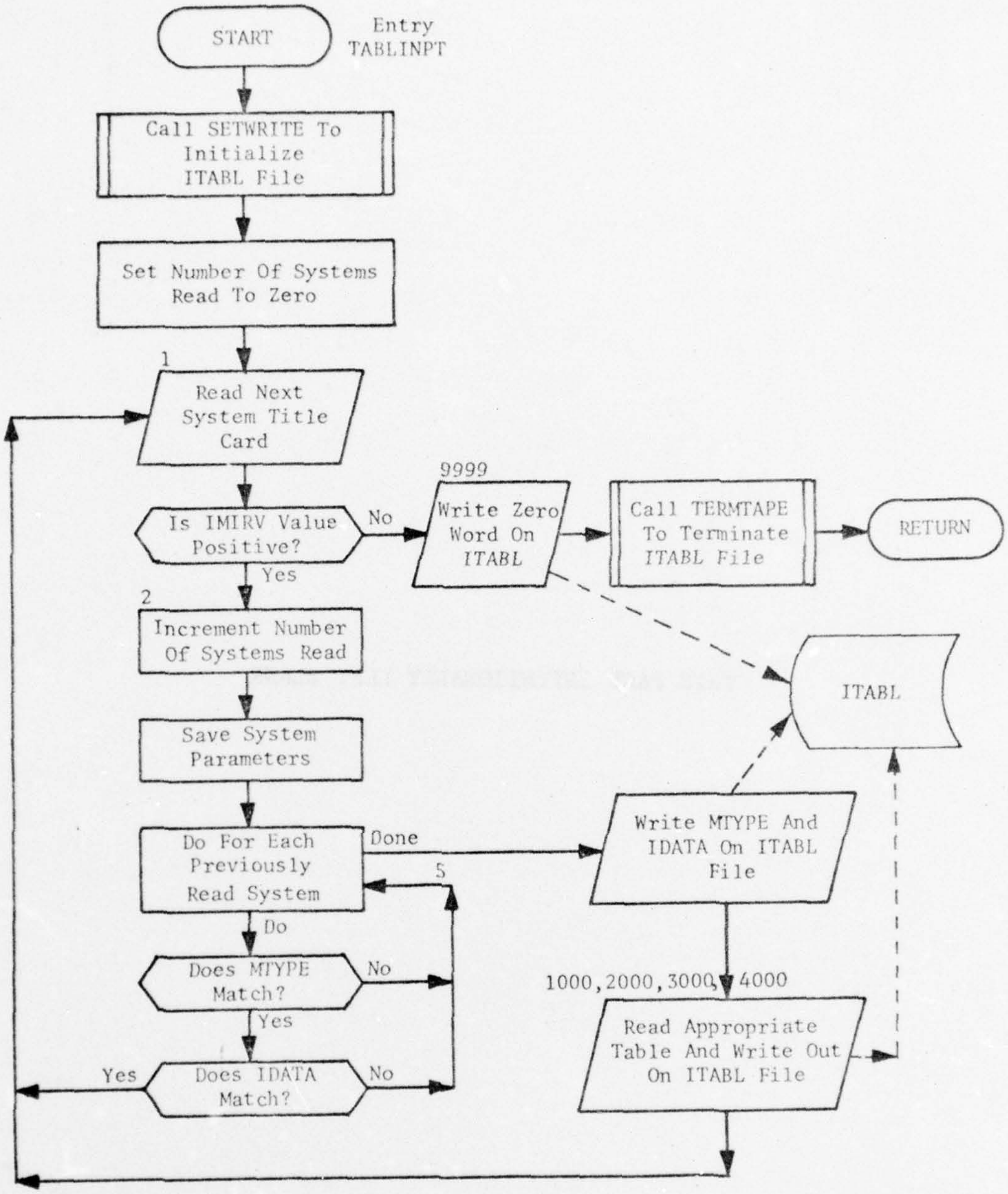


Figure 32. Subroutine TABLINPT (Part 1 of 2)
Part I: Entry TABLINPT

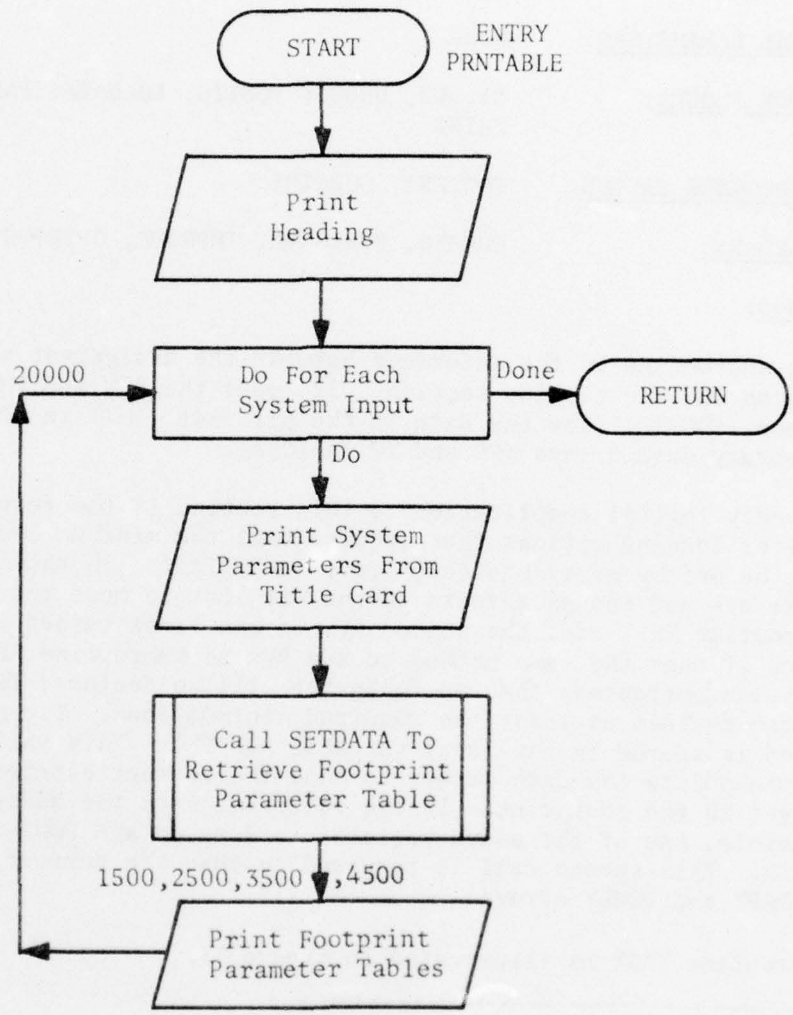


Figure 32. (Part 2 of 2)
Part II: Entry PRNTABLE

2.6.25 Subroutine TEST

PURPOSE:	This routine sets up the test arrays in common /FOOTIO/ for footprint testing.
ENTRY POINTS:	TEST
FORMAL PARAMETERS:	None
COMMON BLOCKS:	C1, C3, DEBUG, FOOTIO, LOADATA, PARAMETR, POTENT, PRINT
SUBROUTINES CALLED:	FOOTEST, GOPRINT
CALLED BY:	CHKSEQ, FUELSAVE, IMPROVE, OPTBOOST

Method:

This subroutine is the interface between the assignment section of the program and the testing section. It loads the RIN and THIN arrays in common /FOOTIO/ from the data in the hit list (IHIT in /POTENT/) and the temporary data arrays (RP and TP in /C3/).

The only logical complication to this routine is the result of the booster loading options that require that the minimum load constraint must be met by every booster; i.e., LOADOPT≥3*. In this case, if there are not enough targets in the hit list to meet the requirement, subroutine TEST adds the needed RVs to the first target in the list. Since it uses the same method to add RVs as subroutine ADDRV, this addition guarantees that no footprint will be declared feasible unless it can contain at least the required minimum load. The number of RVs added is stored in the local variable NOFFSET. This variable is used to manipulate the data arrays to show the correct entries for each real target in the footprint. If the footprint with the added vehicles proves feasible, one of the added vehicles is removed and FOOTEST is called again. This second call is required to load the correct values in the DELRAFT and TOFLY arrays.

Subroutine TEST is illustrated in figure 33.

---

* Input value = MNLREQ

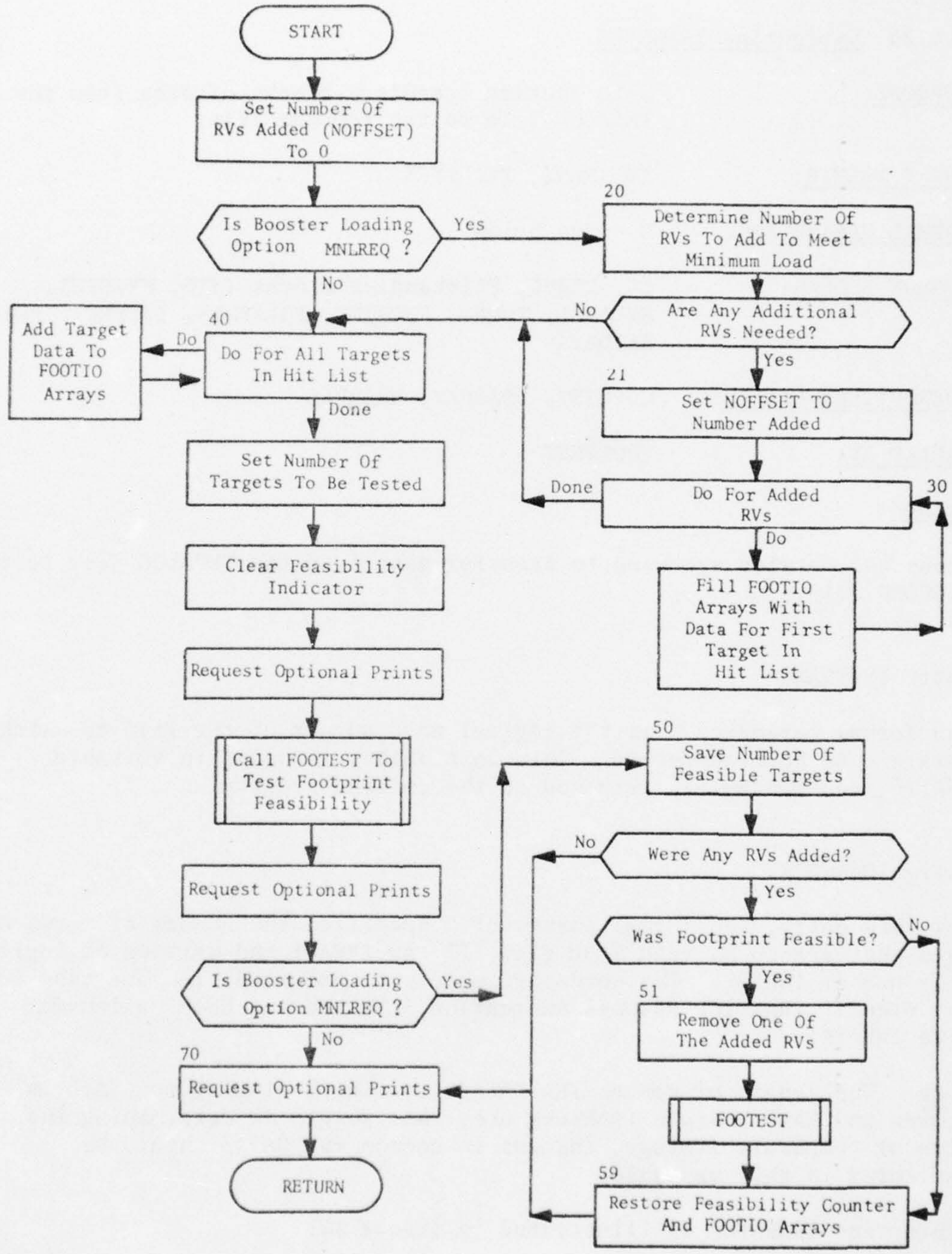


Figure 33. Subroutine TEST

2.6.26 Subroutine TRANSFER

PURPOSE: This routine transfers blocks of data from the TMPALOC file to the ALOCGRP file.

ENTRY POINTS: INITRANS, TRANSFER

FORMAL PARAMETERS: N - See below

COMMON BLOCKS: C4, DEBUG, Filehandler Blocks (ITP, MYIDENT, MYLABEL, TWORD, NOPRINT, FILABEL), IFTPRNT, PRINT, RAIDATA

SUBROUTINES CALLED: GOPRINT, RDARRAY, WRARRAY

CALLED BY: FOOTPRNT

Method:

These two entries are used to transfer data from the TMPALOC file to the ALOCGRP file.

Entry INITRANS

The formal parameter N is the logical unit number of the file to which data are to be transferred. This unit number is saved in variable IWRITE, and control is returned to the calling program.

Entry TRANSFER

For this entry, the formal parameter N specifies the number of words of data that are to be read from file ITP (or IREAD) and written on logical file number IWRITE. The words are merely transferred from one tape to the other. TRANSFER assumes subroutine SETWRITE has been called for file IWRITE.

Note: The length of common /RAIDATA/ from the beginning to LRAID is stored in LRAID. Since TRANSFER uses this length in determining the size of temporary storage, changes in common /RAIDATA/ should be reflected in this variable.

Subroutine TRANSFER is illustrated in figure 34.

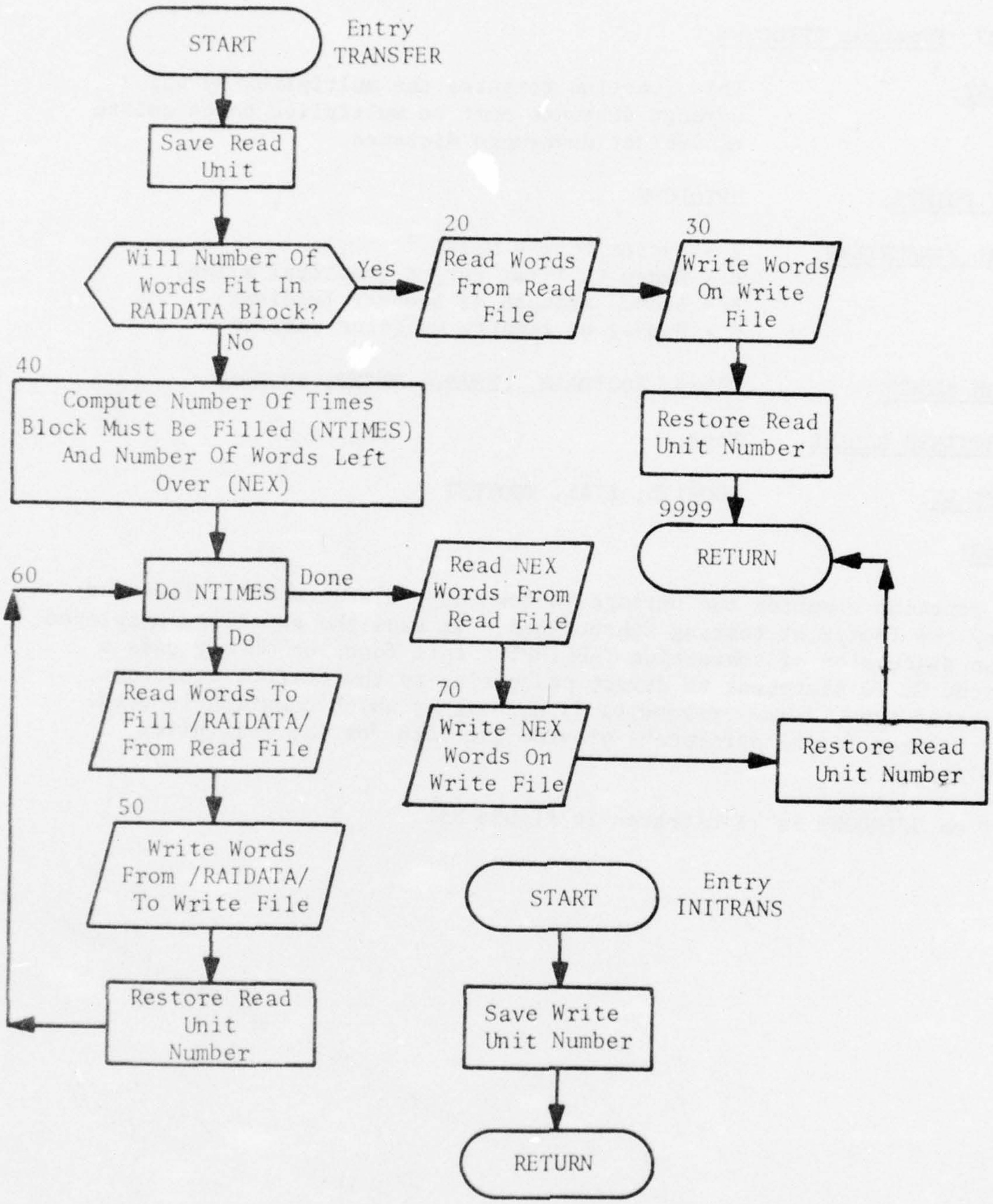


Figure 34. Subroutine TRANSFER

2.6.27 Function UPTODOWN

PURPOSE: This function computes the multiplier by which uprange distance must be multiplied to calculate equivalent downrange distance.

ENTRY POINTS: UPTODOWN

FORMAL PARAMETERS:
I - System type - MTYPE
R - Range to first target (nautical miles)
AZ- Launch azimuth of booster (radians)
N - Number of reentry vehicles carried

COMMON BLOCKS: CSYS4, FOOTDATA, PENADD, PRINT, SHRTDAT

SUBROUTINES CALLED: None

CALLED BY: BOOSTIN, EVAL, FOOTEST

Method:

This function computes the uprange-to-downrange distance multiplier for use by the footprint testing subroutines. It uses the equations displayed in the discussion of subroutine TABLINPT. This function merely uses a computed GO TO statement to direct processing to the correct equation. The system type (formal parameter I) determines which equation is used. The remaining formal parameters provide the data for the multiplier calculation.

Function UPTODOWN is illustrated in figure 35.

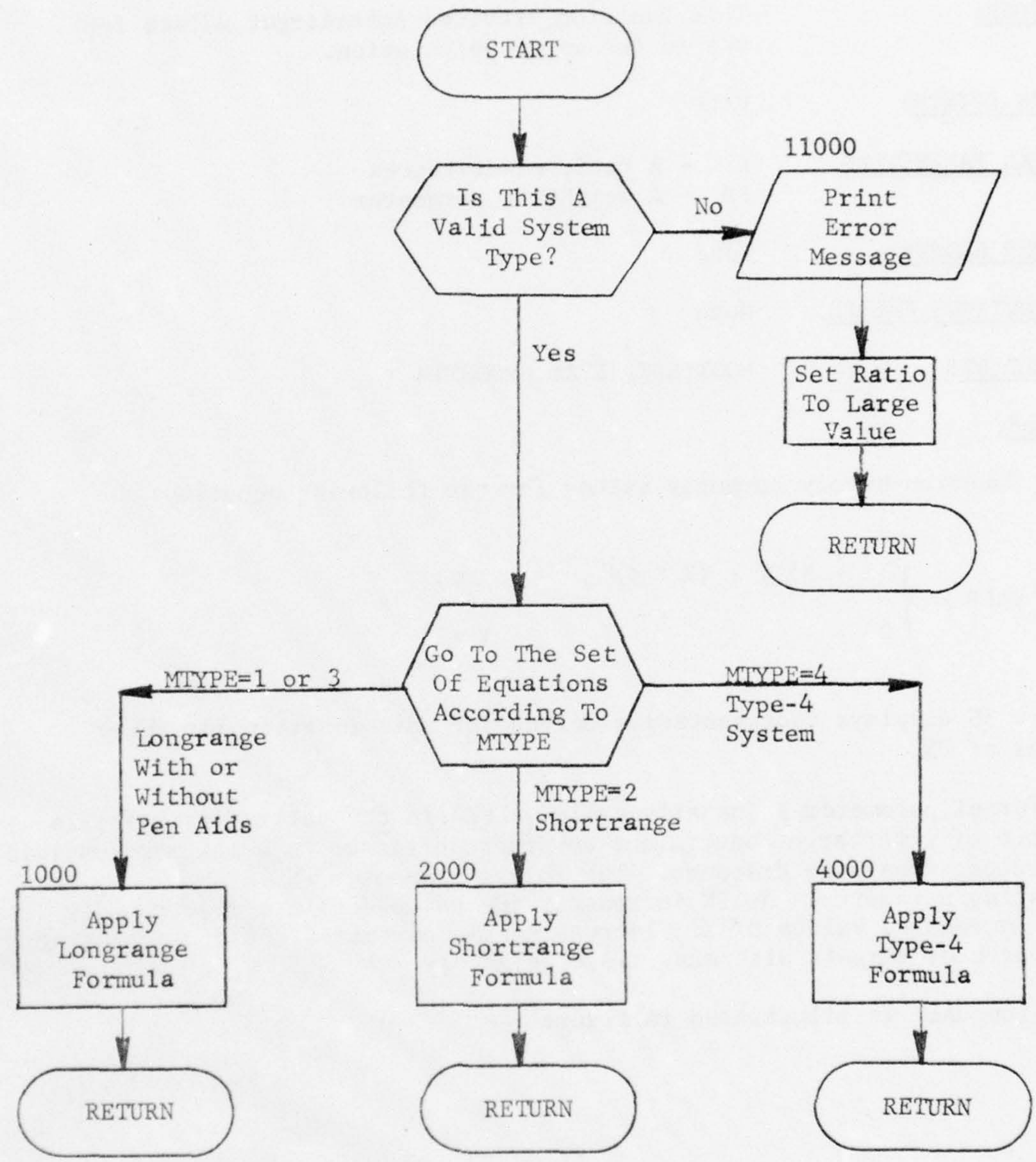


Figure 35. Function UPTODOWN

2.6.28 Function VALF

PURPOSE: This function provides intertarget values for use in the worth calculation.

ENTRY POINTS: VALF

FORMAL PARAMETERS: X - A ratio of distances
FN - A weighting parameter

COMMON BLOCKS: None

SUBROUTINES CALLED: None

CALLED BY: FOOTPRNT, EVAL, BOOSTIN

Method:

This function merely computes values for the following equation:

$$VALF = \begin{cases} (1 - X)/(1 + (X * FN)) & X \leq 1 \\ 0 & X > 1 \end{cases}$$

Figure 36 displays representative curves for this function for three values of FN.

The formal parameter X (usually called ALPHA in the calling program) is a ratio of intertarget equivalent downrange distance to a maximum feasible equivalent downrange distance. The formal parameter FN is a weighting parameter. As FN increases, the value declines more rapidly with increasing values of X. Increasing FN has the effect of increasing the worth of targets with many close neighbors.

Function VALF is illustrated in figure 37.

Table 10. STRKFILE Format (Missile Record)*
Written From Array EVTDATA

WORD	DESCRIPTION
1	ISORTN (Sortie Sequence Number)
2	Side
3	Command and control index
4	Group index
5	Plan delay (alert or non alert)
6	Payload index
7-9	Zero
10	Missile type
11	ICLASS=1
12	Launch region
13	Alert status
14-17	Zero
18	Number of missiles
19	Number of targets
20-37	Missile indices
38-55	Site indices
56-73	Target indices
74-91	Offset latitude
92-109	Offset longitude
110-127	Flight times in hours
128-145	Weapon site latitude
146-163	Weapon site longitude
164-181	Target latitude
182-199	Target longitude
200-217	Designator code of target
218-235	Task and country owner codes of target
236-253	Country code of target
254-271	Flag code of target
272-289	Missile salvo number
290	Logical Array (of 18 elements) containing height of burst for each strike

*The first word output on the STRKFILE is TARFAC; missile and bomber records follow.

Table 11. STRKFILE Format (Bomber Record)*
Written From Common OUTSRT
(Part 1 of 2)

WORD	DESCRIPTION
1	ISORTN (Sortie Sequence Number)
2	Sortie index
3	Group index
4	Corridor index
5	Vehicle index
6	Refuel index
7	Depenetration index
8	Payload index
9	Base index
10	Weapon type
11	Base latitude
12	Base longitude
13	Number of targets
14-23	Type of target
24-33	Latitude of target
34-43	Longitude of target
44-53	Latitude of weapon offset
54-63	Longitude of weapon offset
64-73	Index of target
74-83	Designator code (DESIG) of target
84-93	Task and country owner codes of target
94-103	Country code of target
104-113	Flag of target

*The first word output on the STRKFILE is TARFAC; missile and bomber records follow.

Table 13. (Part 7 of 9)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
/C3/* (cont.)	DESIG(1030)	Target designator codes
	TASK(1030)	Target task and country owner codes
	CNTRYLOC(1030)	Target country location codes
	FLAG(1030)	Flag codes for targets
Input** record from ALOCGRP or TMPALOC file	NT	Total number of targets assigned to group
	JGROUP	Group number
	JCORR	Corridor number (=0)
	INDEXNO(1130)	Index numbers of targets (negative if first target assigned to booster)
	TLAT(1130)	Target latitude (degrees)
	TLONG(1130)	Target longitude (degrees)
	INTOT(1130)	Not used
	RVAL	Relative value of strike
	DLAT(1130)	Offset latitude (degrees)
	DLONG(1130)	Offset longitude (degrees)
	DESIG(1130)	Target designator code
	TASK(1130)	Target task and country owner codes
	CNTRYLOC(1130)	Target country location code
	FLAG(1130)	Flight time code
	FTIME	Flight time matrix
	EVTDATA(288)	Missile record as output to STRKFILE (see discussion of STRKFILE output for redefinition of this array)
	LXLFIXMI(32)	Logical array indicating missile fixed assignment
	ISAL(1130)	Salvo number
	DUM(1566)	Unused

* As used when processing a bomber record
** As used when processing a missile record

Table 13. (Part 8 of 9)

OUTPUT DATA FOR STRKFILE*

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
/SORTNO/	ISORTN	Sortie Sequence Number
/OUTSRT/	IOUTSRT	Sortie index
	MYGROUP	Group index
	MYCORR	Corridor index
	INDVEH	Vehicle index
	JREF	Refuel index
	JDPEN	Depenetration index
	KPAYLOAD	Payload index
	LNCHBASE	Base index
	ITYP	Weapon type
	BASELAT	Base latitude
	BASELONG	Base longitude
	NHAP	Number of targets
	HAPTYPE(10)	Type of target
	OBLAT(10)	Latitude of target
	OBLONG(10)	Longitude of target
	DLAT(10)	Latitude of weapon offset
	DLONG(10)	Longitude of weapon offset
	IOBJECT(10)	Index of target
	DSIG(10)	Designator number of target
	TSK(10)	Task and country owner codes of target
	CNTRLC(10)	Country code of target
	FLG(10)	Flag of target
	ATTROUT(10)	Local attrition
	SURVOUT(10)	Cumulative survival probability

*The bomber records only are written from common block /OUTSRT/and /SORTNO/; the missiles are handled separately.

Table 14. (Part 2 of 2)

WORD OF EVTDATA	DESCRIPTION	EQUIVALENCED TO:
217-234	Task and country owner codes of target	KTASK(18)
235-252	Country code of target	KCNTRYLC(18)
253-270	Flag code of target	KFLAG(18)
271-288	Salvo number	KSAL(18)
289	Logical array containing weapon height of burst	LXMYHOB(1)

Table 15. Program POSTALOC Internal Common Blocks
(Part 1 of 16)

BLOCK	VARIABLE OR ARRAY*	DESCRIPTION
/ARAYSIZE/	MBASEPG	Maximum number of bases (or squadrons) per group (150)
	MC	Maximum number of corridors (30)
	MT	Maximum number of targets per group (1,100)
	ML	Maximum separate defended zones per entry (three)
	MSTRK	Maximum strikes per sortie (10)
	MSORTY	Maximum sorties per group (100)
	MSRT	Maximum sorties per group per corridor (100)
	MFLY	Maximum number of points in JHIT and IFLY lists (13)
	MAXPA	Maximum number of points allowed by array size (25)
/CHGPLN/		(In effect, part of calling sequence for subroutine CHGPLAN)
	JDO	SORTYTGT index for target to be added or deleted by CHGPLAN
	IAIM	SORTYTGT index to flight point for ASM launch
	ISCAN	Controls number of sortie points scanned by EVAL routines
	JAFT	SORTYTGT index to target to precede insertion
	ATO, OTA, BTO, OTB	Calling parameters for CHGPLAN
/CONTROL/	EPSILON	Set to 1001; used in tests of significance

*Parenthetical values indicate array dimensions. All other elements are single word variables.

Figure		Page

TABLES (PART II)

Table 18. Bomber Record, STRKCHNG File
(Part 1 of 2)

WORD	DESCRIPTION
1	Sortie sequence number
2	STRKCHNG indicator information record
3	Sortie sequence number*
4	Sortie index
5	Group index
6	Corridor index
7	Vehicle index
8	Refuel index
9	Depenetration index
10	Payload index
11	Base index
12	Weapon type
13	Base latitude
14	Base longitude
15	Number of targets
16-25	Type of target
26-35	Latitude of target
36-45	Longitude of target
46-55	Latitude of weapon offset
56-65	Longitude of weapon offset
66-75	Index of target
76-85	Designator code (DESIG) of target
86-95	Task and country owner code of target
96-105	Country code of target
106-115	Flag of target
116-125	Local attrition
126-135	Cumulative survival probability
136	Low-altitude range (precorridor legs)

*This record appears if STRKCHNG indicator is nonzero.

Table 18. (Part 2 of 2)

WORD	DESCRIPTION
137	Low-altitude range (before first target)
138	Low-altitude range (after first target)
139	Speed at low altitude
140	Speed at high altitude
141	Range of vehicle without refueling
142	Range of vehicle with refueling
143	Delay before takeoff
144	Regional index
145	Alert status
146	Bomber identification
147	Available low-altitude range
148	Range decrement at low altitude
149	Distance to recovery
150	Distance to origin
151-160	Effective distance between target
161-170	Height of burst information
171-180	Change in time information
191-200	Change indicators for targets
201	Height of burst indicator
202	Change of time indicator

Table 19. Missile Record, STRKCHNG File
(Part 1 of 2)

WORD	DESCRIPTION
1	Sortie sequence number
2	STRKCHNG indicator information record
3	Sortie sequence number*
4	Side
5	Command and control index
6	Group index
7	Time of launch
8	Payload index
9-11	Zero
12	Missile type
13	ICLASS=1
14	Launch region
15	Alert status
16-19	Zero
20	Number of missiles
21	Number of targets
22-39	Missile indices
40-57	Site indices
58-75	Target indices
76-93	Offset latitude
94-111	Offset longitude
112-129	Flight times in hours
130-147	Weapon site latitude
148-165	Weapon site longitude
166-183	Target latitude
184-201	Target longitude
202-219	Designator code of target
220-237	Task and country owner codes of target
238-255	Country code of target

*This record is only provided when STRKCHNG indicator is nonzero.

Table 19. (Part 2 of 2)

WORD	DESCRIPTION
256-273	Flag code of target
274-291	Missile salvo number
292-309	Height of burst code
310-327	Delta time information
328-345	Change indicators for targets
346	Height of burst indicator
347	Change of time indicator

Table 26. (Part 2 of 2)

Plan Information Blocks:

One block for each event in plan (regular or refuel abort)

WORD	DESCRIPTION
1	Time increment since last event
2	Place index
3	Event type
4	Latitude of event
5	Longitude of event
6	Offset latitude } for weapon delivery
7	Offset longitude } for weapon delivery
8	Warhead index
9	Damage expectancy
10	Cumulative time to event

Target Information Block:

One block for each weapon delivery

WORD	DESCRIPTION
1	Index number for target
2	Target designator code
3	Target task and country owner code
4	Target country location code
5	Target flag code
6	Height of burst code

Table 27. Format of PLANTAPE Record
(Missile Plans)
(Part 1 of 2)

Header Block:

WORD	DESCRIPTION
1	Sortie sequence number
2	Side
3	Group index
4	Zero
5	Missile record counter
6-7	Zero
8	ICLASS (=1)
9	Missile type (ISIMTYPE)
10	Launch region
11	Alert status
12	Payload index
13	Salvo number
14	Number of missiles
15	Number of targets
16	Time of launch in hours
17-22	Zero
23	Warhead type
24	Missile type (Plan Generator type)
25	Missile function code
26	End sentinel (=4HLAST after last good record; zero otherwise)

Target Information Blocks:

One block for each target in plan

WORD	DESCRIPTION
1	Flight time
2	Site index

Table 27. (Part 2 of 2)

WORD	DESCRIPTION
3	Missile index
4	Target latitude
5	Target longitude
6	Weapon site latitude
7	Weapon site longitude
8	Warhead type
9	Reliability
10	Target index number
11	Target designator code
12	Target task and country owner codes
13	Target country location code
14	Target flag code
15	Height of burst code

Table 28. Format of PLANTAPE Record (Tanker Plans)

Header Block:

WORD	DESCRIPTION
1	Zero (tankers are not assigned sortie sequence numbers)
2	Side
3	Group number
4	Zero
5	Sortie number
6	Base index number
7	Vehicle index number
8	ICLASS = 3
9	Weapon type index
10	Zero
11	Alert status
12-13	Zero
14	Total number of events
15-25	Zero
26	End sentinel (=4HLAST after last good record; zero otherwise)

Plan Information Blocks:

One block for each event in plan

WORD	DESCRIPTION
1	Time increment since last event
2	Place index
3	Event index
4	Latitude of event
5	Longitude of event
6	Cumulative time to event

4.4.3 Output Files from Overlay INTRFACE. INTRFACE produces two output tapes, all containing 80-column BCD card images.

a. The STRIKE tape (PTAPE) containing strike card ("S" card) type data for each missile and bomber weapon scheduled for delivery (for bombers, only the weapons associated with the primary plan are considered). The strike card format is shown in table 29.

b. The sortie specifications tape (ABTAPE). This tape contains a set of BCD card images for each missile, bomber(primary mission), and tanker plan contained on the PLANTAPE. A card set consists of one "A" card which contains general descriptive information and a variable number of "B" cards which define the individual flight legs of the mission. The "A" and "B" card formats are described in tables 30 and 31, respectively.

4.5 Common Block Definition

4.5.1 External Common Blocks. The common blocks used by program PLANOUT in processing input/output (I/O) files are grouped by overlays. Those used by PLANO1, the first overlay are shown in table 32; PLNTPLAN external common blocks are shown in table 33; INTRFACE (external and internal) common blocks are shown in table 34.

4.5.2 Internal Common Blocks. In addition to the common blocks associated with I/O operations, the common blocks used internally by program PLANOUT are given in table 35 for overlay PLANO1, and in table 36 for overlay PLNTPLAN. INTRFACE internal common blocks are included in previously referred-to figure 34.

Table 29. Format of Strike Card on STRIKE Tape (Part 1 of 2)

CARD COLUMN	VARIABLE NAME	INFORMATION		CONTENT
1		Strike card indicator		S
2		Zero		0
3	ICMD	Command or function code		1-9
4-8	ISORTN	Sortie sequence number		00001-99999
9-10	LDAY	Month		01-12
11-12	DAHOMIMO	Day	of target detonation	01-31
13-14		Hour		00-23
15-16		Minutes		00-59
17-18	YEAR	Seconds		00-59
19-20	LATTGT	Degrees	Latitude of target	
21-22		Minutes		
23-24		Seconds		
25		North or South		N or S
26-28	LONGTT	Degrees	Longitude of target	
29-30		Minutes		
31-32		Seconds		
33		East or West		E or W
34-38	JDESIG	Target designator		2 Alpha, 3 Numeric
39-40	IPLS	PLS - Probability of prelaunch survival		-1-99*
41-42	IPTP	PTP - Penetration probability		-1-99*
43-44	IWSR	WSR - Weapon system reliability		-1-99*
45	IREG	Region code		
46-48	IFRAC	Fission/yield ratio		000-999
49		Blank		
50-54	IYIELD	Weapon yield (KT)		00001-99999
55-57	KHOB	HOB (Height of burst in hundreds of feet)		000-999

*A value of -1 implies 100 percent probability

Table 29. (Part 2 of 2)

CARD COLUMN	VARIABLE NAME	INFORMATION	CONTENT
58-60	KCEP	CEP (in 100s of feet)	000-999
61-62	ITASK	Target task code	2 Alpha
63-64	ICNTRY	Code for country of target location	2 Alpha
65-66	ICOWN	Code for country of target owner	2 Alpha
67-68	IPABORT	Percent chance of target attrition	00-99
69	ISCP	RV number	1-9
70-71	IPLNETYP	Plane type code	01-99
72-73	IWPNTYPE	Weapon type code	01-99
74-77	IUNIT	Unit number	0001-9999
78-79	ISORTIE	Sortie number	01-99
80		Blank	

Table 30. Format of "A" Card on Sortie Specifications Tape (ABTAPE)

CARD COLUMN	VARIABLE NAME	INFORMATION	CONTENT
1		Card designator	A
2-4	LINEAI	Line number	001-999
5-8	IUNIT	Unit number	0000-9999
9-10	ISORTIE	Sortie number	01-99
11-12		Blank	
13-14	IPLNETYP	Plane type code	01-99
15		Zero	0
16-17		Blank	
18		Zero	0
19-22	IREFTIME	Reference time (launch time in hours and minutes)	0000-9999
23	ITIMEREF	Time reference	1 = launch
24-30		Zero	0000000
31		Blank	
32-34	LANPLTYP	Plane type Mnemonic	3 Alpha
35		Blank	
36-37	LCNTRY	Country Code of launch base	2 Alpha
38		Blank	
39-40	ICMD	SAGA Vehicle-Function Code	1=ICBM 2=IRBM 3=MRBM 5=SSB/SSBN 6=SSGN 7=LRA 0,4,8,9 not used
41-80		Blank	

Table 31. Format of "B" Card on Sortie Specifications Tape (ABTAPE) (Part 1 of 3)

CARD COLUMN	VARIABLE NAME	INFORMATION	CONTENT
1		Card designator	B
2-4	LINEB	Line number	001-999
5-8	IUNIT	Unit number (QUICK index number)	001-9999
9-10	ISORTIE	Sortie number	01-99
11-12	LEG	Leg number	01-99
13-14	IOP	Event or operation type indicator	1 - Takeoff 2 - Aerial refueling 3 or 4 - Dogleg 5 - Not used 6 - ASM launch 7 - ASM on target 8 - Decoy release 9 - Decoy impact 10 - Missile or bomb on target 11 - MIRV on target 12 - Not used 13 - Recovery if bomber. Splash if air breathing missile. 14 - Splash (Ballistic Missiles)

Table 31. (Part 2 of 3)

CARD COLUMN	VARIABLE NAME	INFORMATION	CONTENT
15-19	IDES	Location identifier for given operation code. The contents column shows the entry associated with the following codes.	
		IOP= 1	Base index INDEXNO
		= 2	Area number
		= 3	Zeros
		= 4	Zeros
		= 6	00001
		= 7	Target DESIG Code
		= 8	00001
		= 9	00001
		=10	Target DESIG Code
		=11	Target DESIG Code
		=13	Recovery base INDEXNO if bomber
20-25	LAT	Latitude at end of leg	(Degrees, minutes, seconds)
26-33	LON	Longitude at end of leg	(Degrees, minutes, seconds)
34	MODE	Mode of operation	1 - High altitude 4 - Low altitude
35		Zero	0
36-41	ICUMTIME	Time of event	Hours, minutes, and seconds
42	ISOUTH	Southern Latitude indicator	S if southern latitude, blank if not

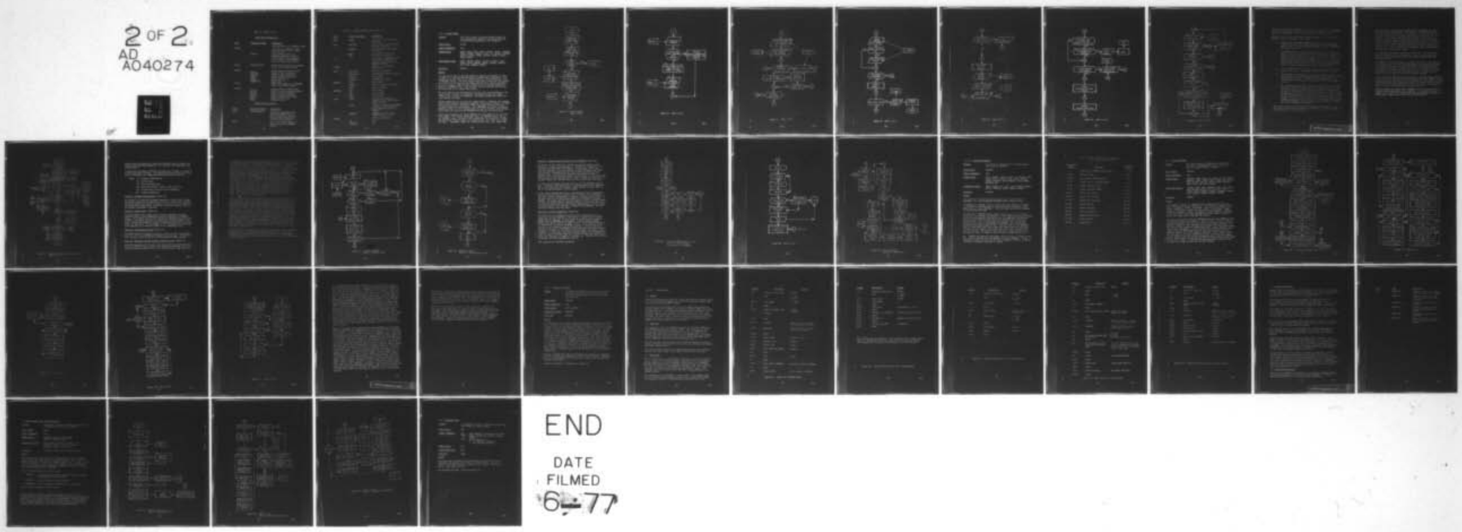
Table 31. (Part 3 of 3)

CARD COLUMN	VARIABLE NAME	INFORMATION	CONTENT
43-44	ISCP	Sequential warhead number	01-10
45		Zero	0
46		Blank	
47-49	CAZIM	Launch/Back Azimuths in degrees	000 - 360
50	IECM	ECM	0 - Off 1 - On
51		Zero	0
52-53	IPAR1	Warhead type	01-99
54	IHXX	Height of burst (HOB)	0 - Ground
55-66		Zeros	1 - Air
57-58	IPLNETYP	Plane type code	01-99
59-60	ICNTRY	Code for country of target location	2 Alpha
61	IREGB	Region code	1-9
62		Blank	
63-64	ITASK	Target task code	2 Alpha
65-67	IHXXX	Height of burst (hundreds of feet)	000-999
68-72	LYIELD	YIELD(MT)	00001-99999
73-75	JCEP	000-999	CEP (100's feet)
76-77	ICOWN	Code for country of target owner	2 Alpha

AD-A040 274 COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON D C F/G 15/6
THE CCTC QUICK-REACTING GENERAL WAR GAMING SYSTEM (QUICK). PROG--ETC(U)
APR 77

UNCLASSIFIED CCTC-CSM-MM-9-74-VOL-4-PT NL

2 OF 2
AD
A040274



END
DATE
FILMED
6-77

Table 33. (Part 6 of 6)

INPUT FROM STRKCHNG FILE

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
		STRKCHNG header (see STRKCHNG format)
/CONTR1/		File control parameters (fully described under Internal Common Block section)
	LREAD(2)	Sortie Sequence Number and STRKCHNG indicator. Bomber or missile plans follow if indicator is nonzero; else STRKFILE plans are used for this sortie
/BLOCK/	LOCK/BLOCK(320)	Missile plan; identical to STRKFILE record (see STRKFILE format)
/OUTSRA/		Change arrays for missile plans
	HOBM(18)	Height of burst information
	DLTAM(18)	Change in time information
	INDRM(18)	Change indicator for targets
	IHOBM	Height of burst flag
	ICTIMEM	Change time flag
	DISTEM (36)	Dummy Array
/OUTSRT/	---	Bomber plan; identical to STRKFILE record (see STRKFILE format)
/OUTSRA/		Change arrays for bomber plans
	DISTE(10)	Effective distance between targets
	HOB(10)	Height of burst information
	DLTA(10)	Change in time information
	INDR(10)	Change indicators for targets
	IOHOB	Height of burst flag
	ICTIME	Change time flag

OUTPUT DATA FOR EVENTAPE

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
BLOCK	LOCK/BLOCK(320)	Initially, the missile plan record from STRKFILE (see STRKFILE format); later, the missile plan record for the EVENTAPE and/or PLANTAPE
INDATA	----	The output EVENTAPE bomber or tanker record (see EVENTAPE format)

Table 34. INTRFACE Common Blocks (Part 1 of 4)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
CUMNO	ICUMNO	Cumulative number of types in each class
DATE	TGTITIME	Time in hours, since beginning of game, of target hit
	DAHOMIMO	BCD; coded day, hour, minute, of target hit
	YEAR	BCD; second of target hit
	LDAY	BCD; month of target hit
DIN		Second part of PLANTAPE record for missiles, bombers, or tankers (see description of PLANTAPE records)
FILABEL		FILABEL is a filehandler common block
FILES	TGTFILE(2)	Not used in INTRFACE
	BASFILE(2)	BASFILE unit number and length
	IDUM(11)	Dummy array
	PLANTAPE	PLANTAPE unit number
FRACYLD	YIELD	Weapon yield
	FISFRAC	Fission fraction for weapon
	NWPN	Number of weapons
GAMETIME	KDAY	Day of game
	KMON	Month of game
	KYEAR	Year of game
IFUNC	IFUNC	IFUNC(I) is function or command to which QUICK plane type I belongs
	JFUNC	JFUNC(I) is function or command code if I is odd; JFUNC(I+1) is the Hollerith name for JFUNC(I)
	INDFUNC	INDFUNC(I) is the same as JFUNC(I) when I is even
INTRHL		Hollerith constants
	HA	6HA
	HBLANKXX	6H

4.6.1 Overlay PLANO1

PURPOSE: This first overlay of program PLANOUT permits the user to make minor changes to the plans generated by QUICK without requiring a new allocation

ENTRY POINTS: PLANO1

FORMAL PARAMETERS: None

COMMON BLOCKS: BASEF, BEGIN, BLOCK, CARDS, CONTROL, CONTR1, CORRCHAR, CORRC1, DPENRE, FILABEL, FILES, ITP, MYIDENT, MYLABEL, NOPRINT, OUTSRA, OUTSRT, PAYDATA, PLNTHL, TARLIST, TWORD, WAROUT

SUBROUTINES CALLED: ABORT, CHGCORR, CHGSRT, DEACTIV, FLTSORT, ITCHG, LOCATE, MKCHG, RDARRAY, RDWORD, SETREAD, SETWRITE, SKIP, SLOG, TERMTAP, WRARRAY

CALLED BY: PLANOUT

Method:

The tape option card is read and printed to initiate processing, as shown in figure 76. This card defines whether PLANTAPE and/or EVENTAPE is to be created and whether this run of PLANOUT will use a recycled version, IST=1, of STRKCHNG, or not. The first sortie change card is read and if the operation code is an "E" there are no changes and the first overlay terminates. If not, the change cards are read, sorted by Sortie Sequence Number and saved on a temporary file. Then the first card is read from the file and the processing of all change cards begins.

The first phase is the initialization of input files and performance of the proper reads. TARFILE is initialized; the header is read into common /TARLIST/ and its file is terminated. Subroutine FINDTAR will reopen and close TARFILE as needed.

BASFILE parameters are now read into common /BASEF/, /CORRCHAR/ and /DPENRE/. /BASEF/ array contains information required if new missile sorties are added; other data are used in subroutine CHGSRT when bomber attrition and low-altitude distance are recalculated. If no STRKCHNG file exists, subroutine CHGCORR is called to fill required arrays for use in FLTSORT. Once defined, these arrays are placed on the STRKCHNG file and kept for future runs.

PLANO1 next determines the latest STRKCHNG file and defines it as the input file, SCHNG1. The various combinations are: no STRKCHNG exists, one or two STRKCHNG exist. Function LOCATE determines the existence of files and subroutine SETREAD stores the creation date and times into common /FILABEL/. Parameters INDSK and INDSKM define input file number and

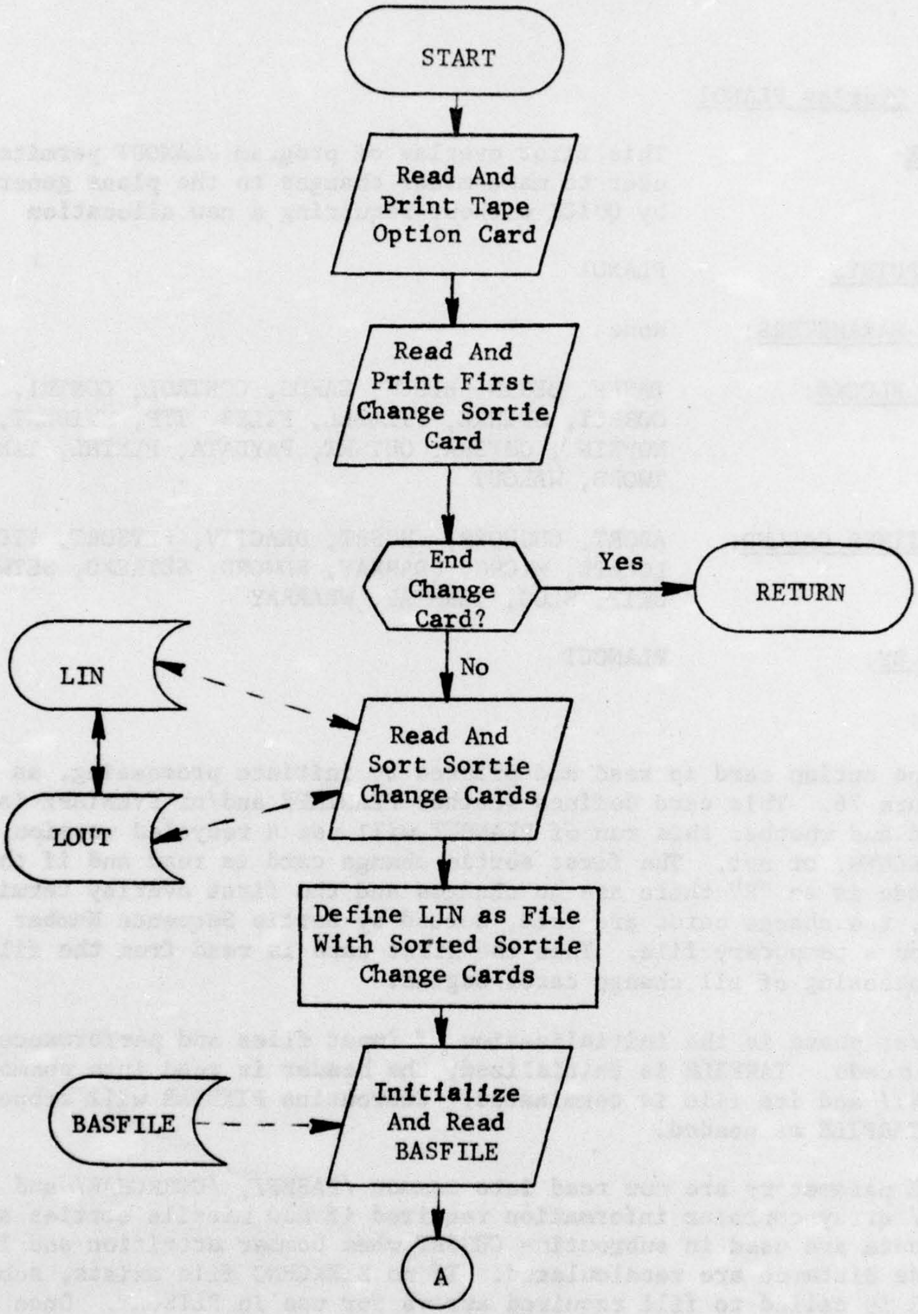


Figure 76. Overlay PLANO1
(Part 1 of 7)

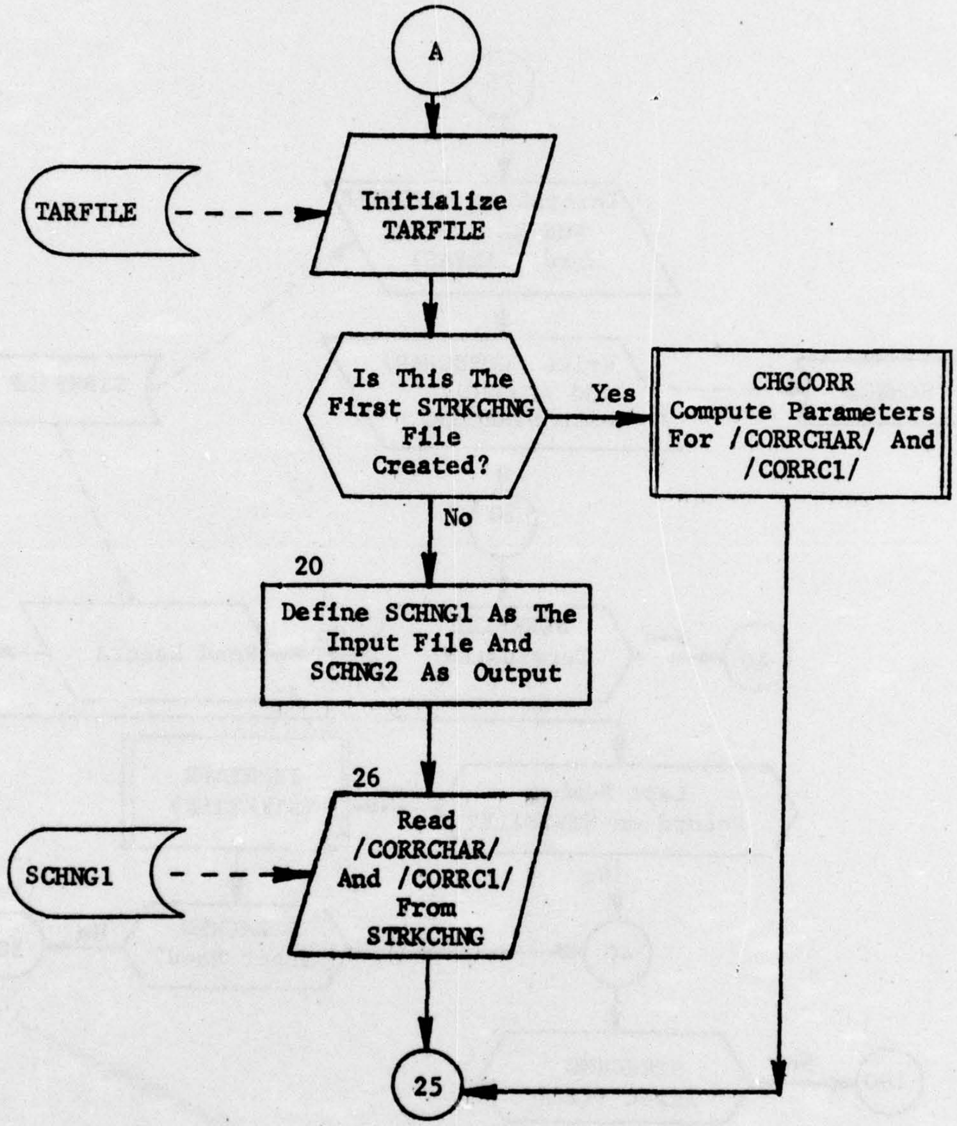


Figure 76. (Part 2 of 7)

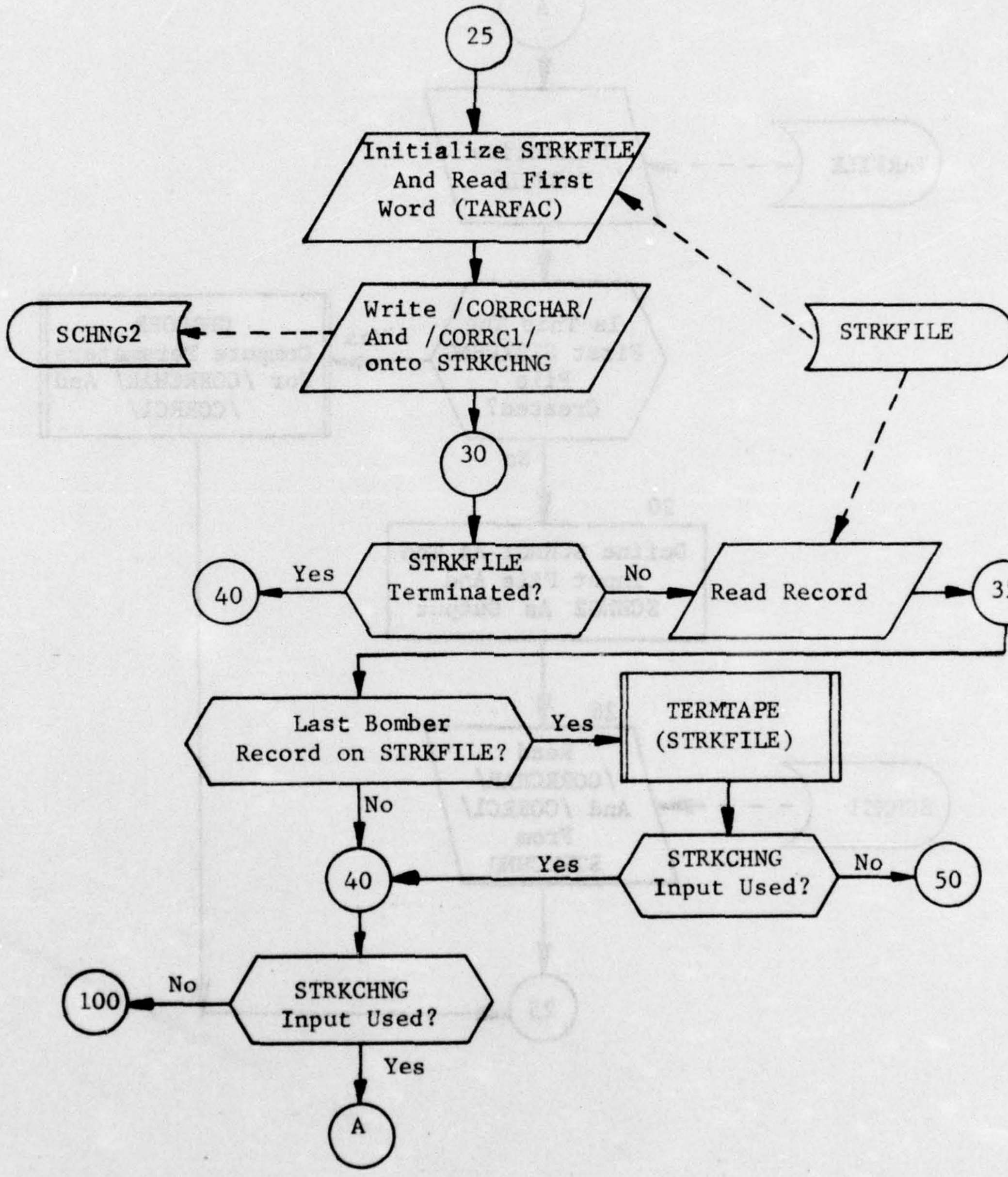


Figure 76. (Part 3 of 7)

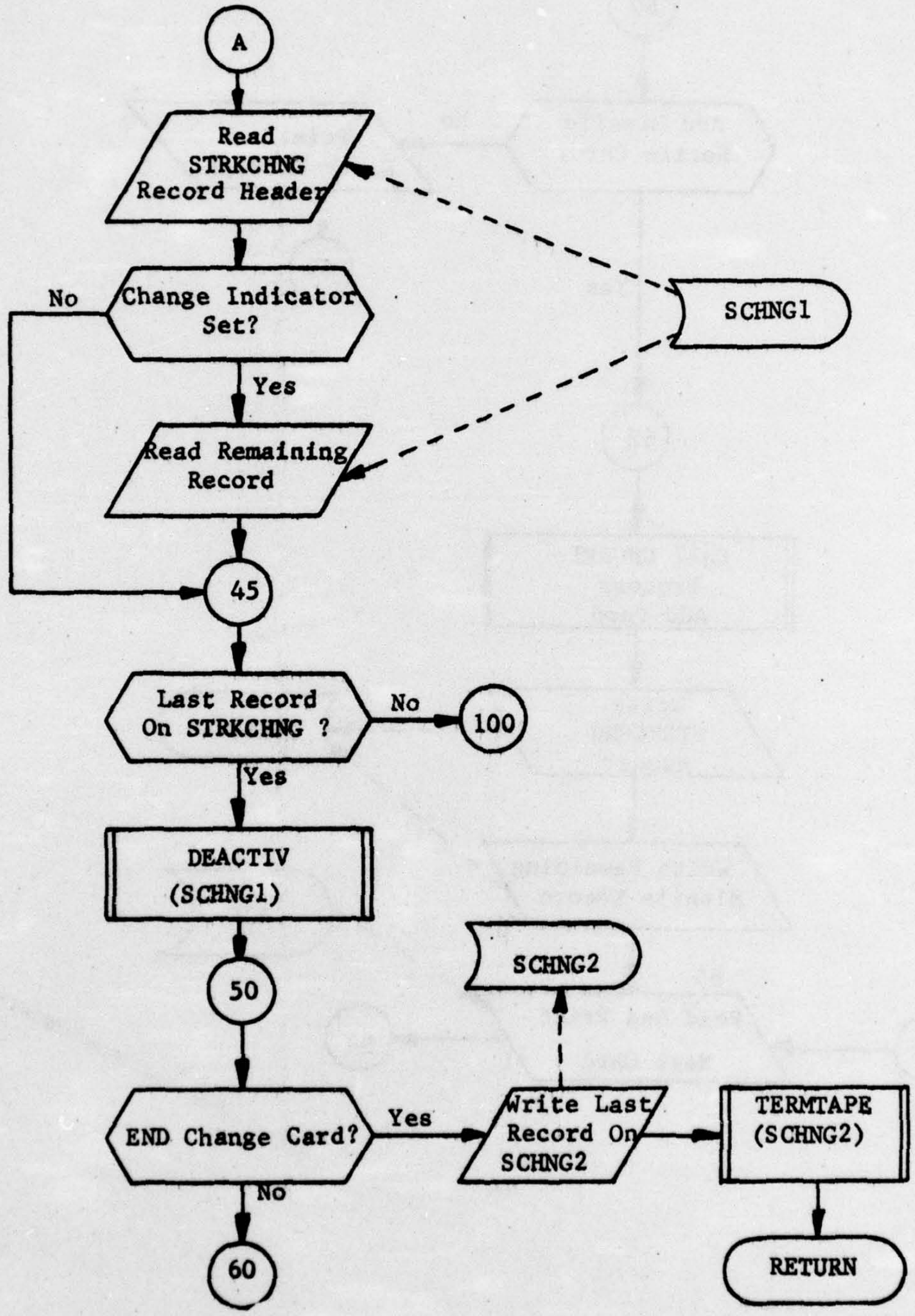


Figure 76. (Part 4 of 7)

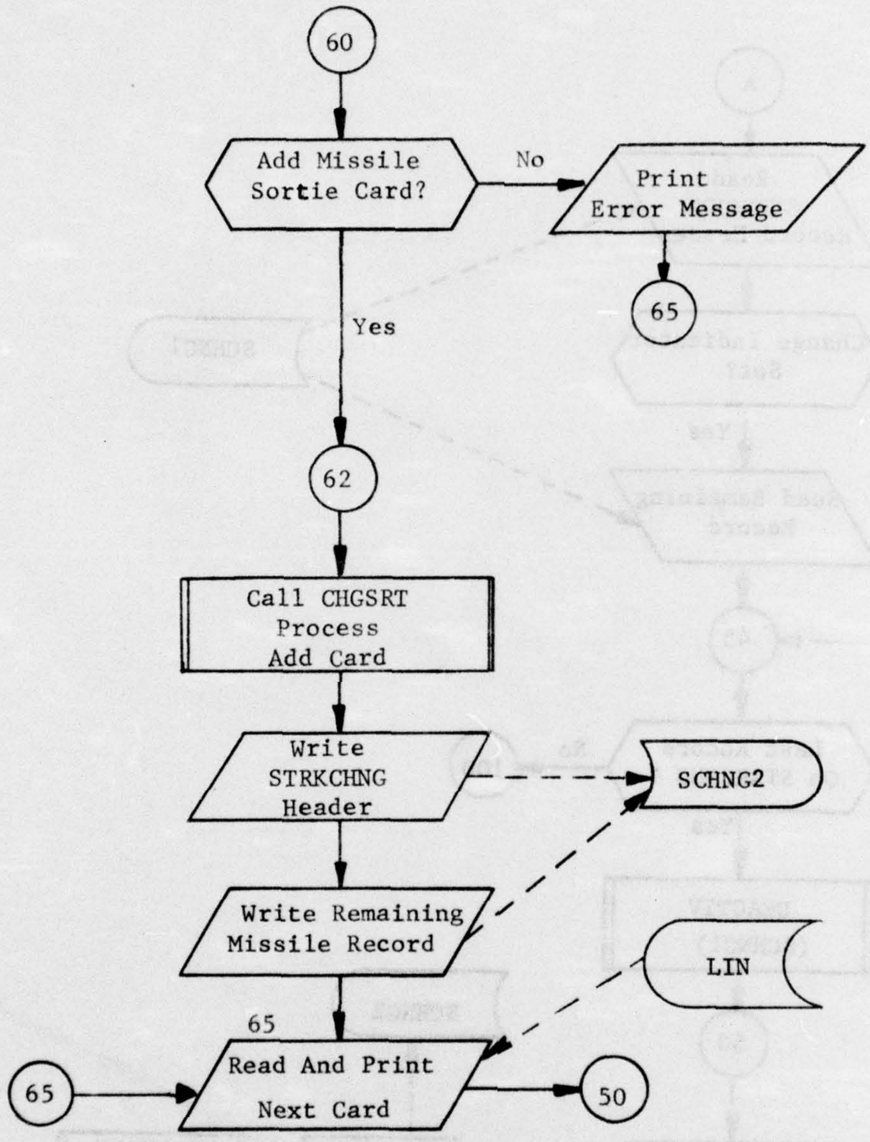


Figure 76. (Part 5 of 7)

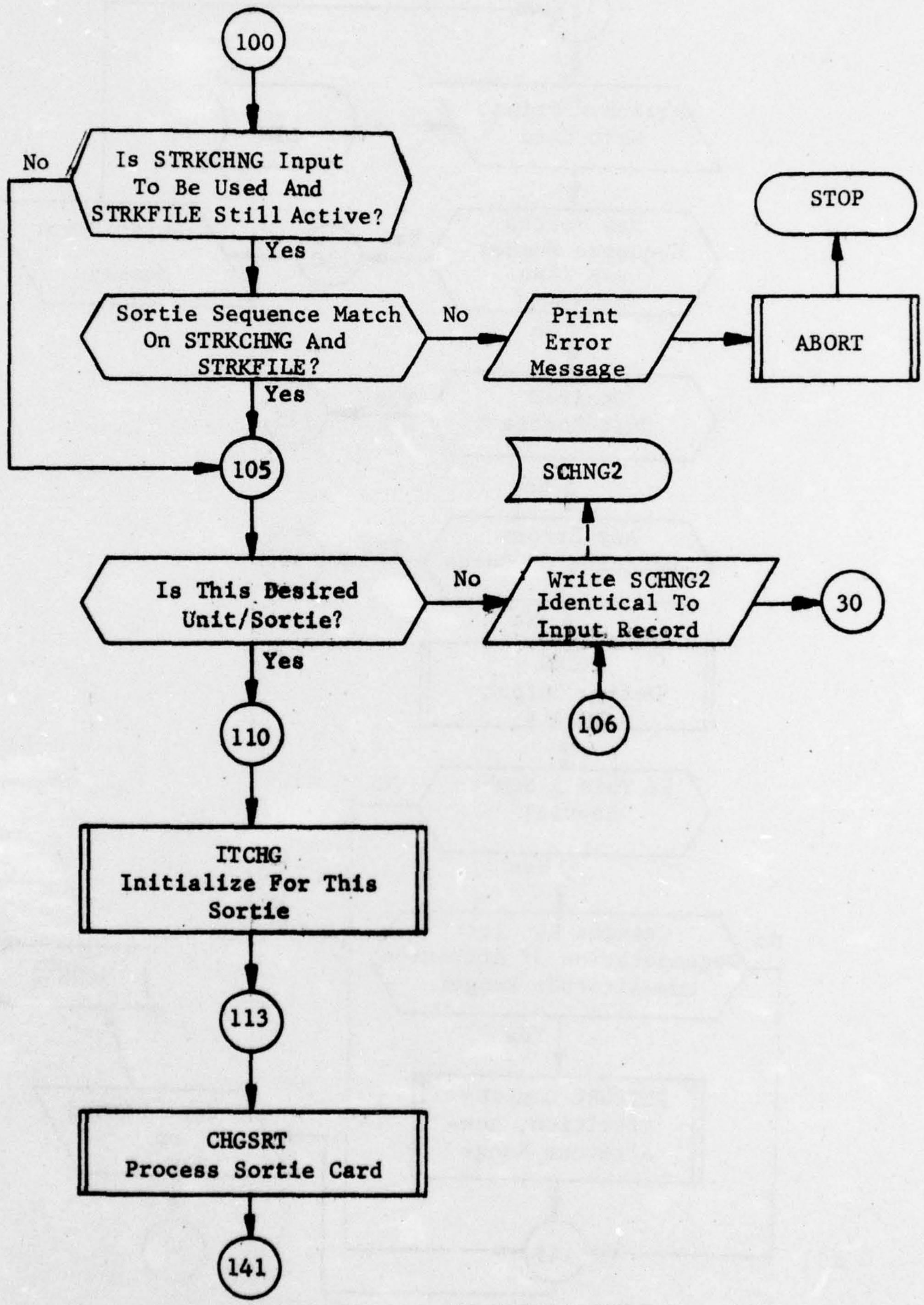


Figure 76. (Part 6 of 7)

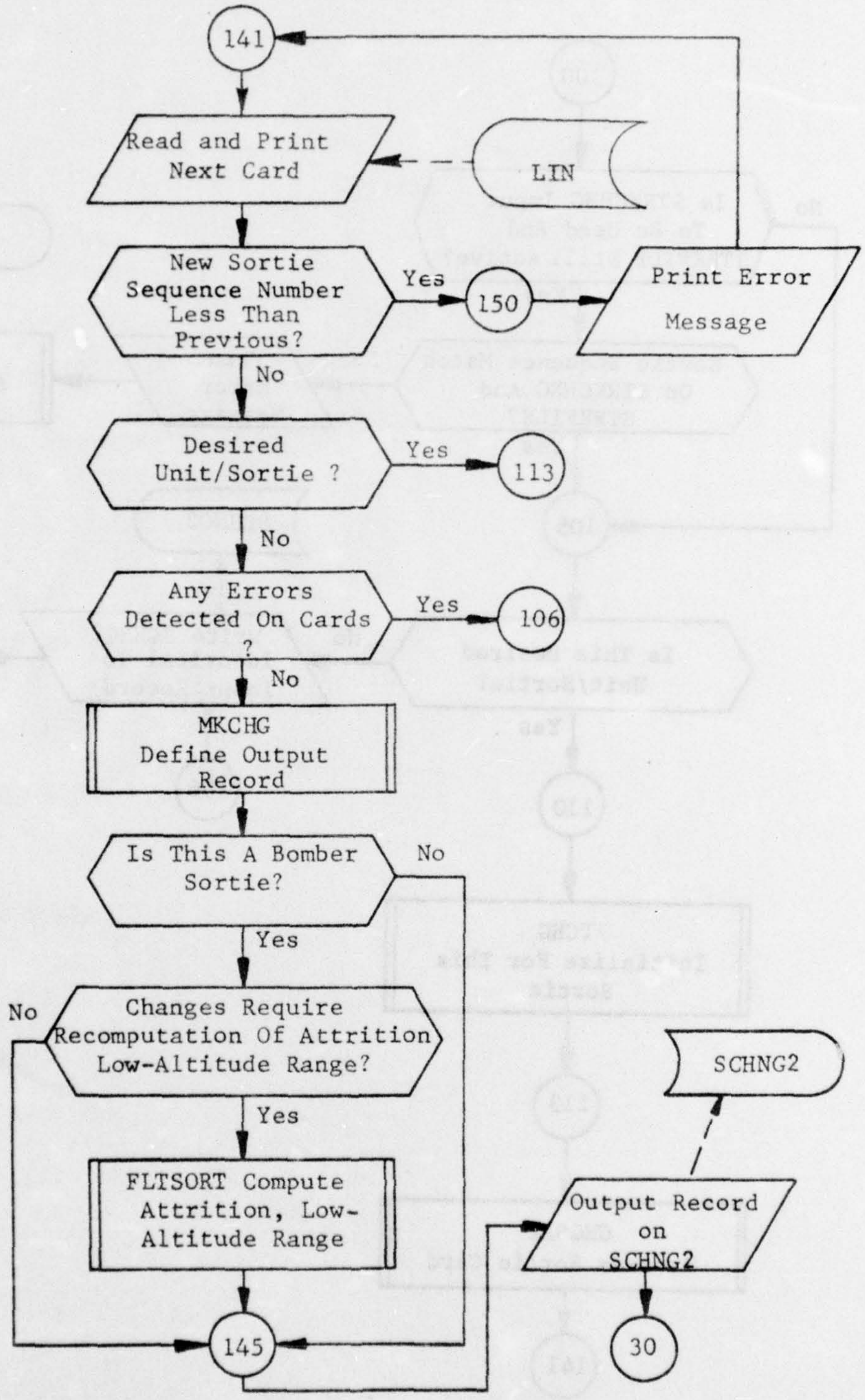


Figure 76. (Part 7 of 7)

file name; IOUTDSK and IOUTDSKM the output number and name. The STRKCHNG header is read, if any file exists. Finally, the STRKFILE is opened and the first word TARFAC is read. SCHNG2 writes the output header.

Merging of input data with STRKFILE and/or STRKCHNG (if it exists) now begins. The cyclic process is as follows:

- Read a plan into common /OUTSRT/ and if it is a missile plan transfer the plan into /BLOCK/ array and finish reading the missile plan. If local parameter NOREAD1 is nonzero, STRKFILE has been terminated.

- A STRKCHNG record is read if it exists and the user desires it, IST≠0. Each read is accomplished in three steps. A two-word header exists for each sortie. If the second word, LREAD(2), is zero the plan as given on the STRKFILE is used; if not this plan has been altered in previous PLANOUT runs. For this case the altered plan follows on the STRKCHNG. The plan is read, again, into common /OUTSRT/ or /BLOCK/, thus eliminating the STRKFILE record. Added information is read into common /OUTSRA/.

- It is imperative that the sortie sequence number on both files match for all plans. A mismatch causes the program to abort.

- The STRKCHNG (or STRKFILE) sortie sequence number, ISORTN, is now compared against the sortie sequence number, ISSN, defined on the last change card read. If ISSN is greater than ISORTN, the present plan is unaltered on this run and written onto SCHNG2 as read.

- For a change request, ITCHG is called to initialize; then CHGSRT to process the sortie change card. A new change card is read and sortie sequence numbers are again compared. Whenever ISSN is greater than ISORTN the present plan may be finalized; MKCHG accomplishes this. If any errors were detected on cards, ICHANGE <0, the output STRKCHNG will be the unchanged input record. Subroutine FLTSORT is used for bomber plan and when recalculation, ICHANGE >0, of attrition is desired. After meeting these checks, the altered SCHNG2 is written and the cycle continues by reading a new plan on STRKFILE and STRKCHNG.

This cyclic process continues until the STRKFILE end-of-file is read. At this stage non-MIRV missile sorties may be added. One sortie change completely defines a new missile sortie.

Figure 83 shows a typical path a bomber would take between the time of its launch and its recovery. The bomber is launched from a base, flies to a refuel point or area if refueling is called for, then to a corridor entry point. It may then fly one or more prespecified doglegs (called precorridor legs) which define a penetration route before reaching the point labeled Corridor Origin. From the origin it flies over the target area and its assigned targets in their proper order. It then enters the depenetration corridor which may also consist of one or more doglegs. From there it flies to the recovery point or base. At any point after the corridor entry it may cross a boundary line between defense zones.

This path may logically be divided into four parts: (1) the launch and refuel portion, (2) the precorridor legs, (3) the target area which is the main part of the plan, and (4) the depenetration and recovery portion.

In PLNTPLAN, each bomber sortie is processed in much the same order as it is flown; that is, first the precorridor section events are posted, then those of the target section, and finally, the depenetration and recovery section events. Besides the posting of the target events themselves, the main processing consists of posting events for changes of altitude, zone crossings, and decoy launches. All postings for bomber events are made in the arrays of common /DINDATA/.

After each bomber plan has been evaluated, and its data stored, PLNTPLAN reads in the next STRKFILE record and executes subroutine CHGCHK. If the new record is the alternate plan for the sortie just processed, the alternate plan for the sortie just processed, the alternate events are posted before the EVENTAPE or PLANTAPE is written. Otherwise, the completed plan is output, and processing begins on the new plan.

After processing all bomber plans, STRKCHNG must be investigated for any missile sorties added by the user in PLAN01. If the sortie sequence number is less than 99999, sorties were added and PLANTMIS processes all added plans. These sorties do not exist on the STRKFILE.

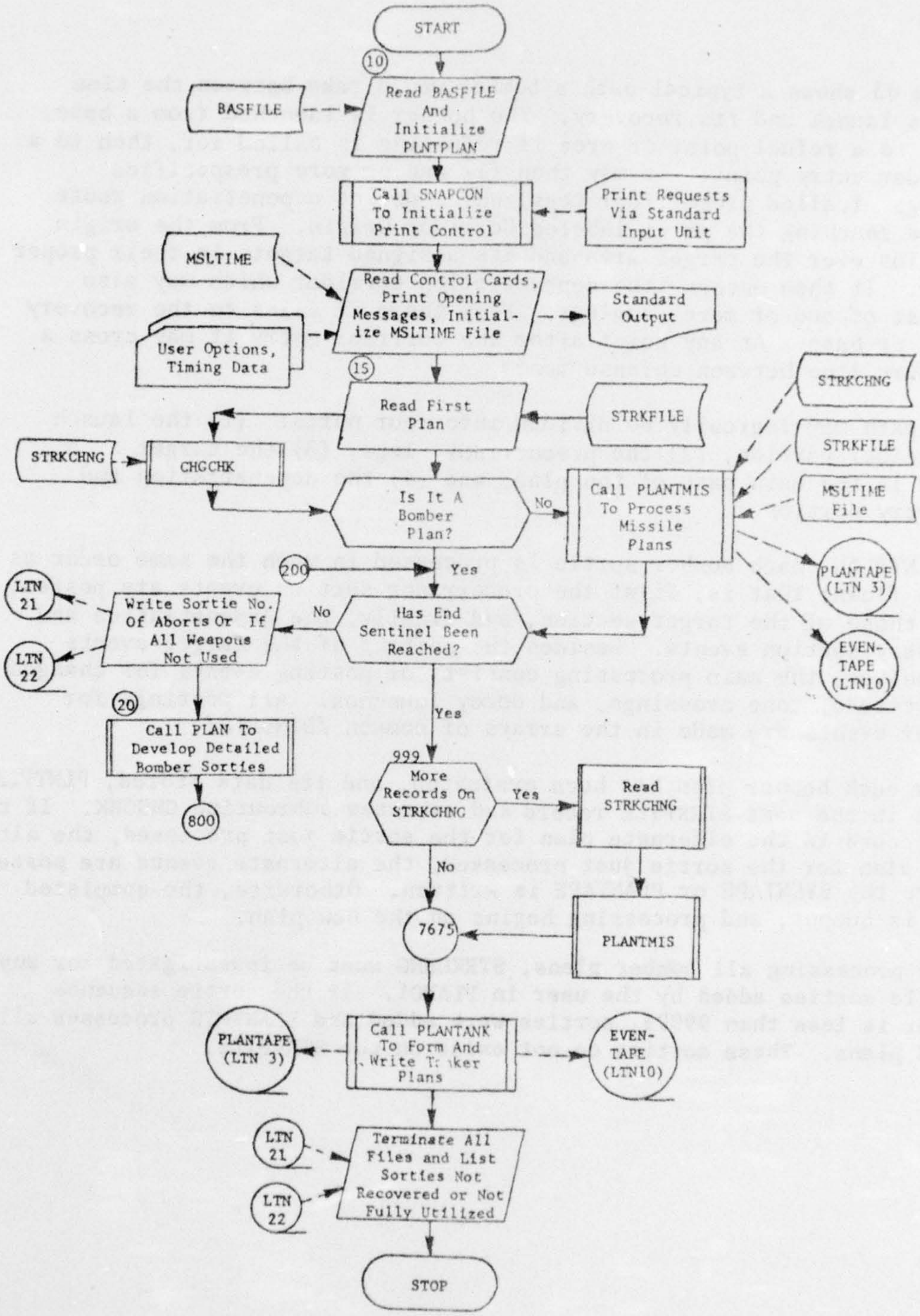


Figure 82. Overlay PLNTPLAN (Macro Flowchart) (Part 1 of 2)

Tanker plans are generated by subroutine PLANTANK after all bomber and missile plans have been completed. All files are then terminated, and PLNTPLAN exits.

To facilitate discussion, PLNTPLAN is divided into "blocks" of coding as noted in the macro flowchart. The remaining description as well as the detailed flowcharts are organized around these blocks, which are:

BLOCK 10 - Program initialization
15 - Control loop
20 - Call subroutine PLAN
80 - Read next /OUTSRT/ record, convert last one
90 - Process final plan and write on EVENTAPE
100 - Program termination

Block 10: Program Initialization (figure 84)

In addition to initialization program variables, coding block 10 reads all required data from the BASFILE, and all user control cards, calling subroutine SNAPCON for the print request cards and subroutine LNCHDATA for the missile timing cards. Files are initialized, headers read, and preliminary information is printed.

Block 15: Control Loop (figure 85)

The first bomber plan is then read in from the STRKFILE, subroutine CHGCHK executed to read STRKCHNG and the main processing of PLNTPLAN begins. If a missile plan is read, subroutine PLANTMIS receives control. Otherwise, the number of events is checked and the sortie INDEX is computed. Subroutine SNAPCON is called to determine which prints are to be active for this plan. If the program is to be terminated at this sortie (print request 14), a branch is taken to the termination block.

Block 20: Call Subroutine PLAN (figure 86)

The bomber sortie is checked to see if it is able to fly to a recovery base or if its full compliment of weapons were allocated. After this is done, subroutine PLAN is called to develop detailed bomber sorties.

Block 80: Read Next /OUTSRT/ Record, Convert Last One (figure 87)

After the processing of the sortie has been completed as described above, the next /OUTSRT/ record is read in and checked to determine if it contains the alternate plan for the sortie just processed. The details of

the processing at this point are determined by whether the current sortie is the primary plan, its refuel-abort alternate, or whether it is a plan without an alternate. In any case, subroutine SWTCHALT is called to convert the CHANGALT events to GO LOW or GO HIGH events. Subroutine DISTIME then is called to compute distances between events and associated time increments, and subroutine DECOYADD is called to allocate the available decoys. Decoy Launches are now added to the detailed History table by examining each event to see if a launch is to be inserted (indicated if the corresponding word in array ILAUNDEC is nonzero). For low-altitude launches (ILAUNDEC = 0), the actual launch point must be computed. The Decoy Launches are inserted by copying each event into a temporary detailed History table. If a GO HIGH event has a decoy launch indicated, the launch is inserted after the GO HIGH. For all other events with indicated decoy launches, the launch is inserted before the event is copied. Decoy launches are posted by adding the event LAUNDCOY to the event array (JTP) and storing the number of decoys launched (>0) in the array usually reserved for the place index (KPL). The remaining information required in the detailed History table is stored in the normal manner.

Decoys are terminated as the detailed History table is recopied into its original arrays. Each time a high-altitude Decoy Launch event is encountered, the total decoy flight time is computed from the distance in array DISTORE (filled by subroutine DECOYADD) and added to the next <u>odd</u> word in an array (TSTORE) which holds the remaining flight time of <u>all</u> decoys which have been launched but not yet terminated at the time of this event. The number of decoys to be terminated is added to the next <u>even</u> word of TSTORE. As each subsequent event is processed, the time since the last event (HDT) is subtracted from the times in TSTORE. Whenever a decoy has no flight time remaining, a LAUNDCOY event, together with the number of decoys being terminated (stored as a negative number) and other relevant information, is added to the detailed History table. If the bomber depenetrates or aborts while decoys are still flying, the remaining decoys are terminated immediately before the final event. It should be noted that decoys launched at low altitude are not terminated.

If the recently read /OUTSRT/ record was an alternate, control transfers back to block 15. Otherwise PLNTPLAN outputs the final plan to the PLANTAPE if the PLANTAPE option has been exercised. If the EVENTAPE has been requested, control then transfers to coding block 90; otherwise it returns to block 15.

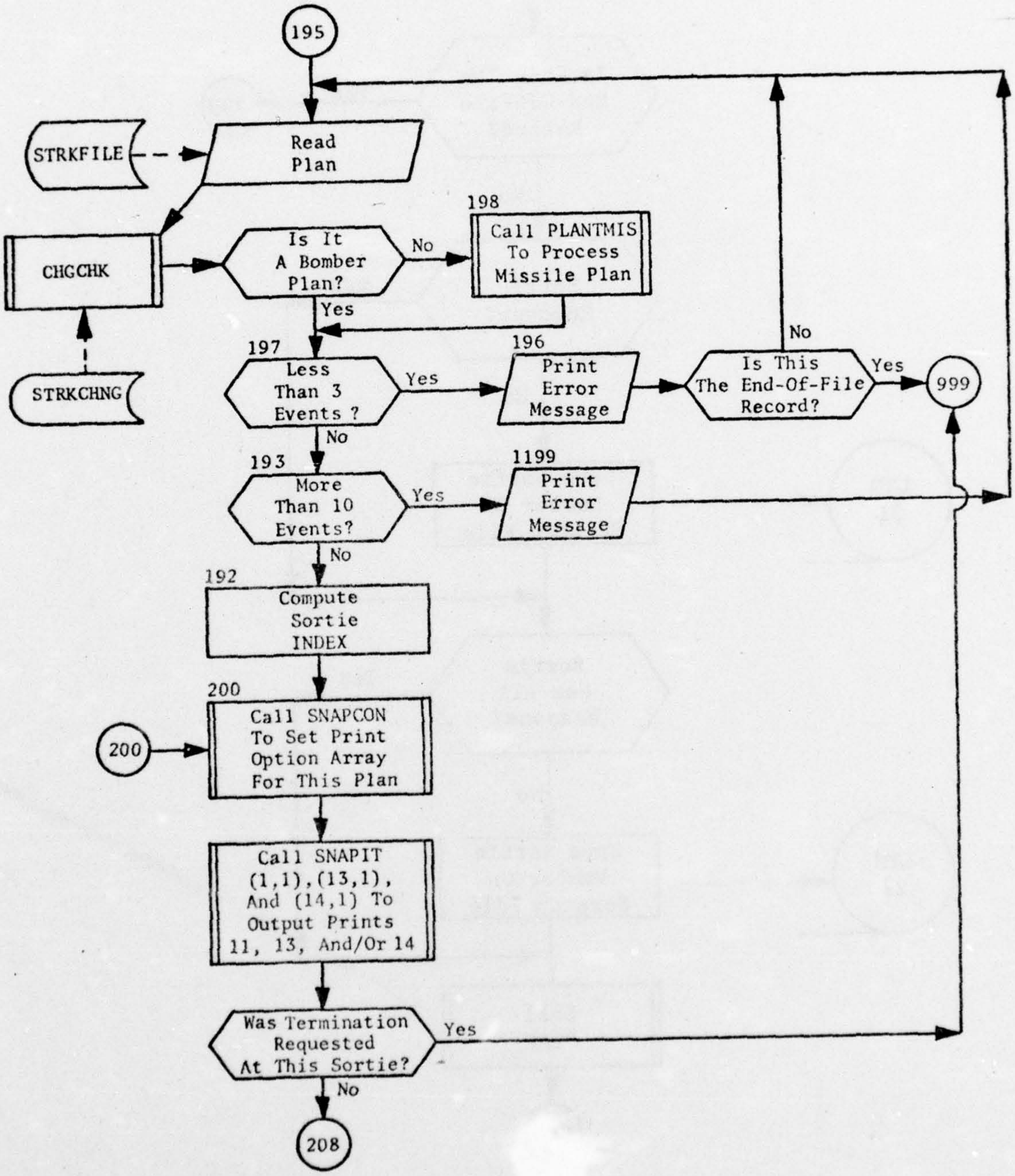


Figure 85. Overlay PLNTPLAN
Block 15: Control Loop

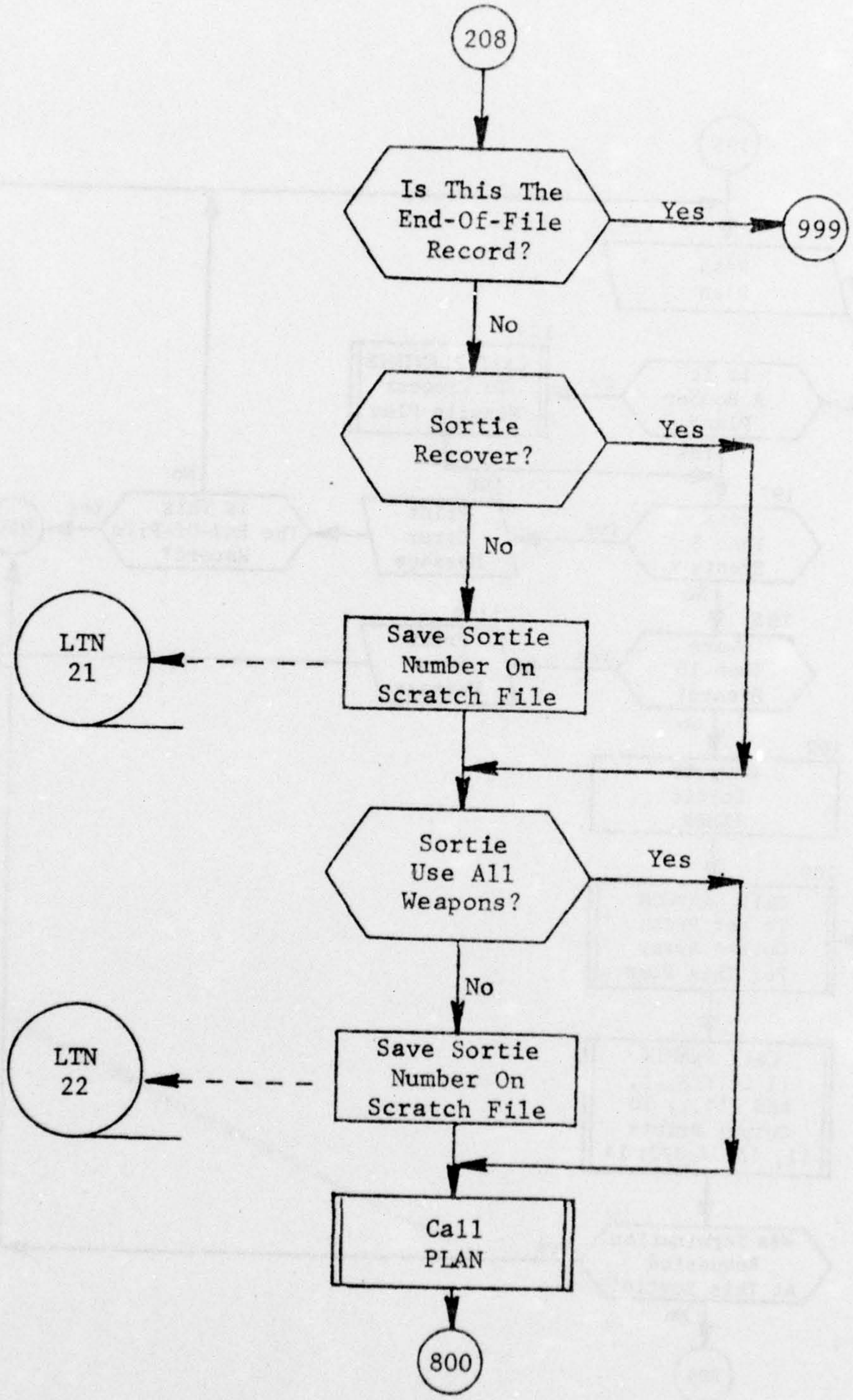


Figure 86. Overlay PLNTPLAN
Block 20: Call Subroutine PLAN

Block 90: Process Final Plan and Write on EVENTAPE (figure 88)

The purpose of this block is to format the plan for output to the Simulator. During this processing, all references to geography (i.e., latitudes and longitudes) are dropped. Dogleg events, which are strictly geographic, are also dropped and time increments associated with them are accumulated with the time increment of the next succeeding event. GO HIGH and GO LOW events are converted back to CHANGALT events and the Weapon table is constructed. (In the event that the plan and its alternate exceed 80 lines, a warning is issued on the standard output unit, and the first 80 lines of this plan are used.)

At this point the height of burst indicator for the EVENTAPE, ZDGZ is set. The target index number (KPL) is compared against the index numbers in the old event list (IBJECO). When a match is found, ZDGZ is set to the desired height of burst MYHOBO.

In the case of weapons which have a time-dependent destruction-before-launch probability (DBL), an extra event is written on the EVENTAPE. This event causes the Simulator to compute a dynamic destruct event for the base using the DBL data table passed on the SIMTAPE by Program INDEXER. The format for this event was shown in table 24 (under OUTPUT). If there are several bombers launched consecutively from the same base, this event precedes only the first bomber LAUNCH event. If two or more of these events for the same base do appear on the EVENTAPE, the Simulator will process only the first.

Block 100: Overlay Termination (figure 89)

The termination of PLNTPLAN occurs when the end sentinel record is reached on the input STRKFILE. This record is identified by a group number greater than 200. At this stage all missile sorties added in PLANO1 onto the STRKCHNG are processed. A sortie sequence number greater than 99999, implies no added sorties. After PLANTMIS evaluates all added plans, subroutine PLANTANK is called to generate and write sorties for all tankers listed on the BASFILE. A sentinel record is written at the end of the EVENTAPE and the PLANTAPE; the bomber recovery information is added to the eventape and the Refuel Area table to the PLANTAPE. At this point, the list of Sortie Numbers of the bombers that could not reach a recovery base and the list of those that did not use their full complement of weapons, is printed out. Finally, all files are terminated, and final messages are printed.

This completes the PLNTPLAN processing.

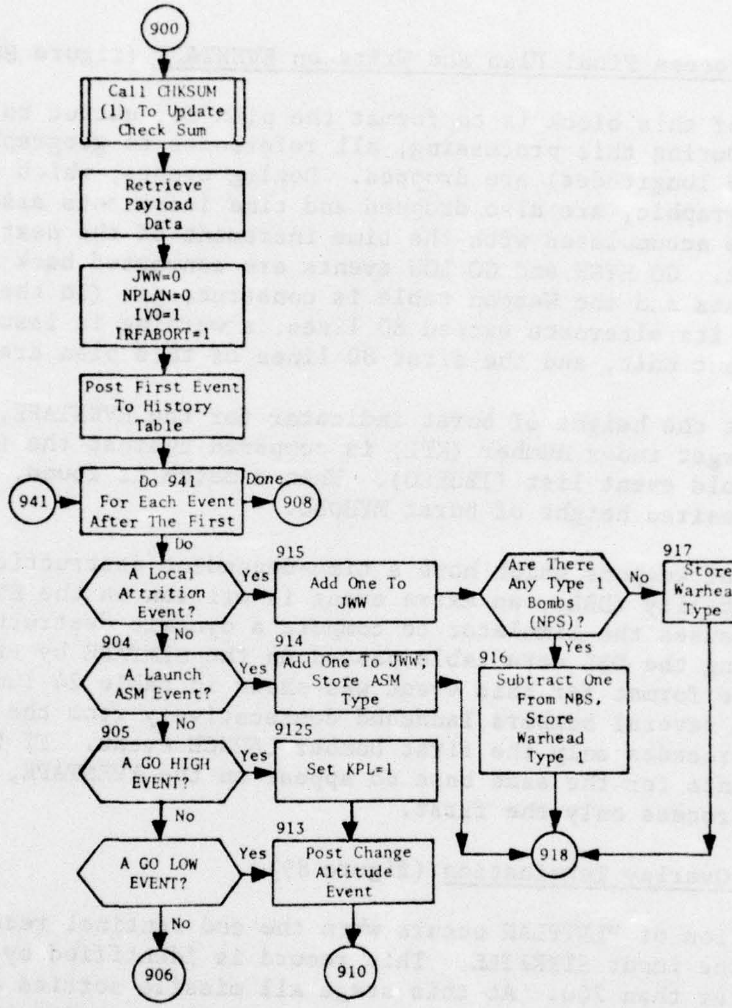


Figure 88. Overlay PLNTPLAN (Part 1 of 4)
Block 90: Process Final
Plan and Write on EVENTAPE

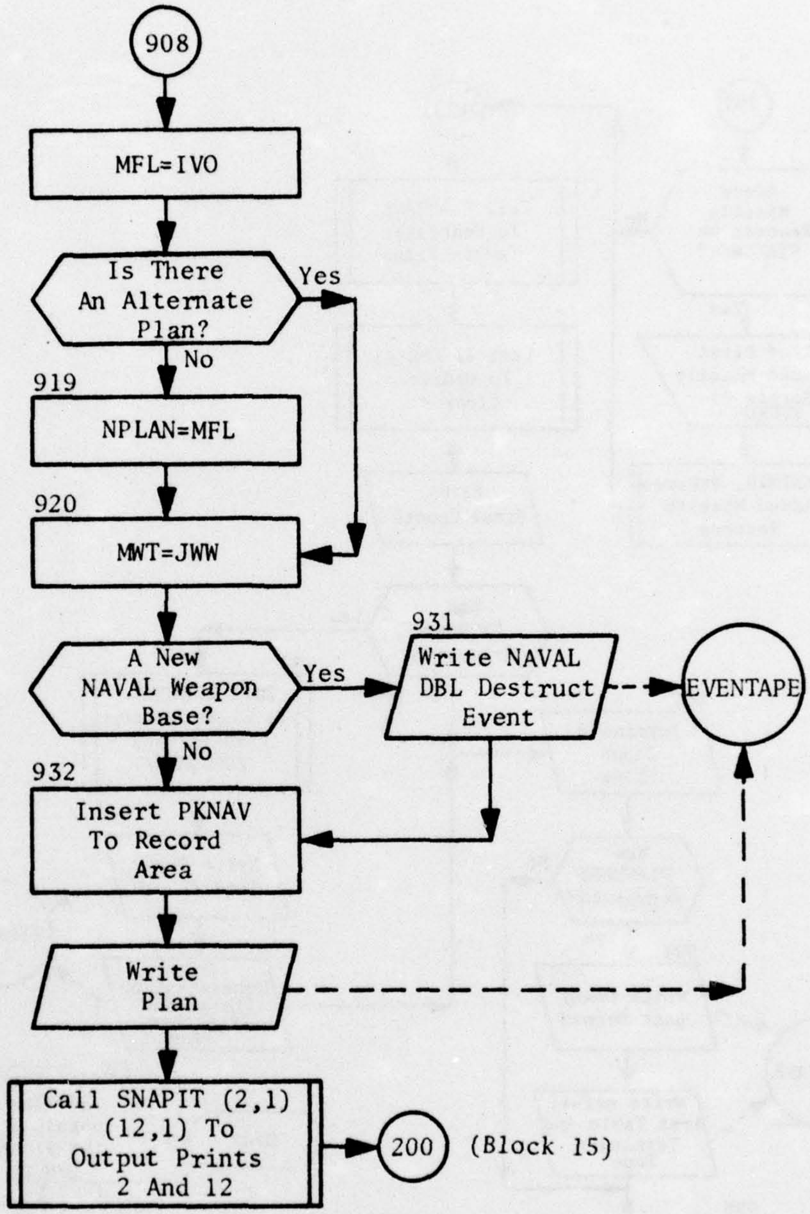


Figure 88. (Part 4 of 4)

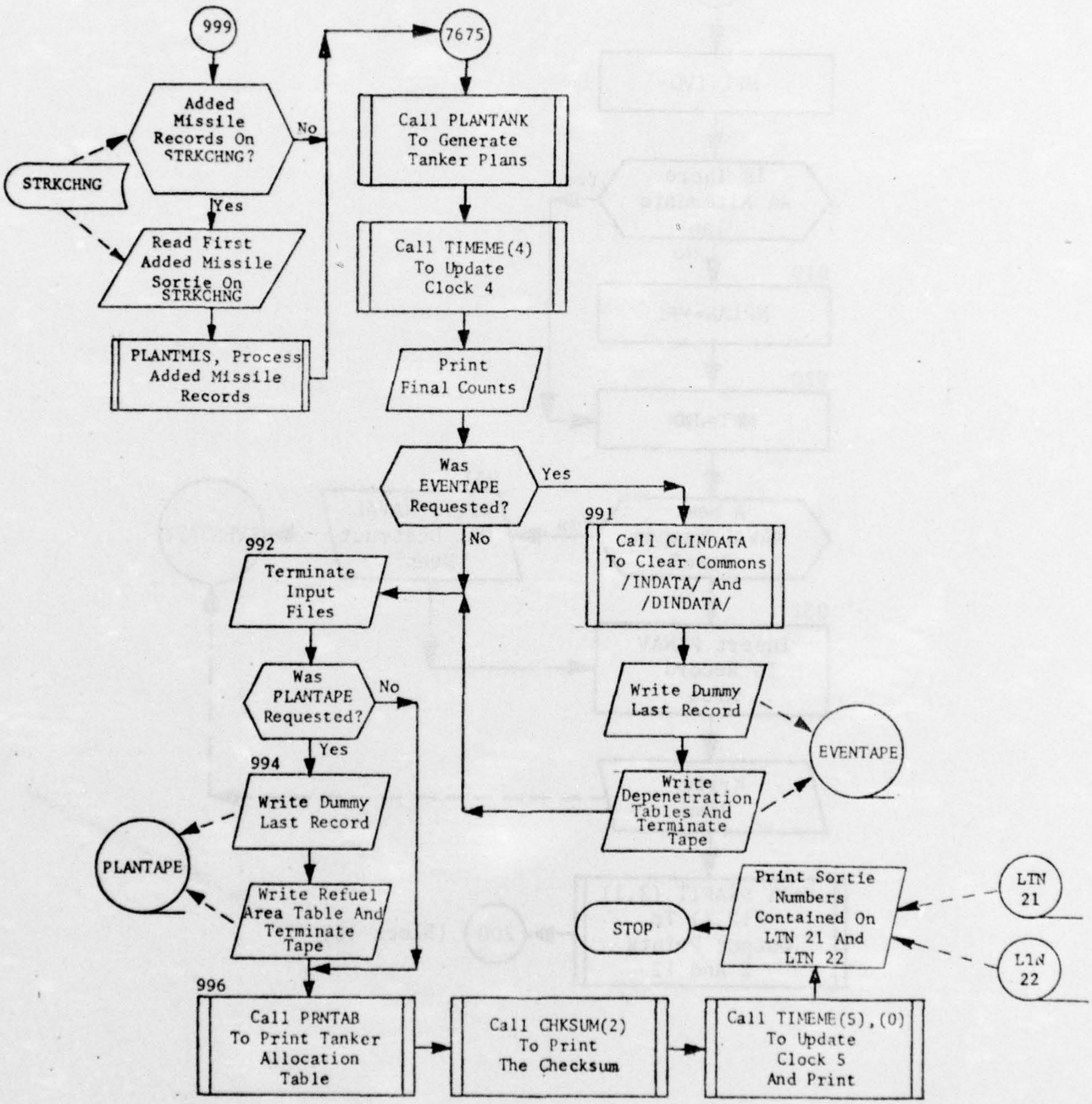


Figure 89. Overlay PLNTPLAN
Block 100: Termination

4.6.2.17 Subroutine PLANTMIS

PURPOSE:	To control the processing of all missile plans input from the STRKFILE.
ENTRY POINTS:	PLANTMIS
FORMAL PARAMETERS:	None
COMMON BLOCKS:	BLOCK, CONTROL, CONTR1, FILES, ITP, KEYLENG, KEYM, MISCT, MRVFLG, NAVAL, OUTSRA, OUTSRT, PAYLOAD, SNAPON, TIMELINE, TWORD, WPNGRPX, DELTA, WAROUT, WPNTYPEX, MAX
SUBROUTINES CALLED:	ABORT, CHGCHK, GLOG, IGET, IPUT, KEYMAKE, RDARRAY, SNAPIT, TIMELNCH, TIMEME, WRARRAY, WRWORD
CALLED BY:	PLNTPLAN

Method:

This subroutine reads and processes missile plans from STRKFILE or STRKCHNG. It is called whenever PLNTPLAN reads a missile record.

The STRKFILE or STRKCHNG record is first moved from /OUTSRT/ to /BLOCK/. To facilitate processing, most of the data is then transferred to the TDATA array in common /TIMELINE/. Table 40 shows the variable placement in arrays BLOCK and TDATA.

The remainder of PLANTMIS merely outputs missile plans in the correct format onto the PLANTAPE and EVENTAPE. If the weapon group under consideration has a time-dependent DBL probability, the subroutine precedes the missile launch events for each base on the EVENTAPE with a naval DBL destruct event. If all events have been deleted from a record that record is not output on the PLANTAPE or EVENTAPE. This situation is identified when target index is zero. The last words in each EVENTAPE missile launch event record are target data words which occur in pairs, one pair for each target. The second word is the time of flight from launch to target. The format of the first word, along with that of the DBL destruct event, is shown in table 24 under PLNTPLAN Output Files.

When PLANTMIS has completed processing a missile record, it reads the next plan from the STRKFILE and calls CHGCHK to read the STRKCHNG. If the information read is another missile record, PLANTMIS processes it without returning to PLNTPLAN; otherwise it returns.

Table 40. Arrays TDATA/ITDATA and BLOCK/LOCK Used in PLANTMIS and TIMELNCH

TDATA/ITDATA INDEX	ATTRIBUTE	BLOCK/LOCK INDEX
---	STRKFILE or EVENTAPE Record words 1-18	1-18
1-18	Missile indices	19-36
19-36	Site indices (from data base)	37-54
37-54	Target indices (from data base)	55-72
55-72	Offset latitude (DLAT)	73-90
73-90	Offset longitude (DLONG)	91-108
91-108	Flight Time (hours)	109-126
109-126	Weapon site latitude	127-144
127-144	Weapon site longitude	145-162
145-162	Target latitude	163-180
163-180	Target longitude	181-198
181-198	Target designator	199-216
199-216	Target task and owner	217-234
217-234	Target country	235-252
235-252	Target flag	253-270

4.6.3 Overlay INTRFACE

PURPOSE: This third overlay of PLANOUT adds information to the output of PLNTPLAN and creates tapes to be used in programs external to QUICK.

ENTRY POINTS: INTRFACE

FORMAL PARAMETERS: None

COMMON BLOCKS: ASMTABLE, CUMNO, DATE, DIN, FILABEL, FILES, FRACYLD, GAMETIME, IFUNC, INTRHL, IPRT, ISOUTH, ITP, MASTER, MODE, MYIDENT, NOPRINT, OFFSYS, PAYLD2, PAYLOAD, PROB1, STUB, TAB, VEHIC, WAROUT

SUBROUTINES CALLED: IGETHOB, IPROB, ITLE, KNOBLANK, LATDT, LATS, LONDT, LONS, NOBLANK, NOFFSYS, NOP, NPLNETYP, NTIME, NUMGET, ORDER, PRNTOFFS, RDARRAY, REORDER, SETREAD, SKIP, TERMTAP, TIMEME, YLDFRAC, FINDTIME

CALLED BY: PLANOUT

Method:

As indicated in figure 141, INTRFACE first reads and prints the user-input cards and stores the information in the appropriate common blocks -- namely, /GAMETIME/, /IPRT/, and /IFUNC/. Then it initialize the file-handler and proceeds to input data for common blocks /MASTER/, /CUMNO/, /PAYLOAD/, /FRACYLD/, /VEHIC/, and /PROB1/ from the BASFILE. Depending on the user print option IPRT, INTRFACE then prints weapon and vehicle tables from common blocks /FRACYLD/ and /VEHIC/.

After this initial input process, INTRFACE processes the PLANTAPE records for missiles, bombers, and tankers -- one at a time. First it reads in a PLANTAPE header block and uses word 8 (missile or weapon type) to obtain LANPLTYP, IPLNETYPE, ICMD, and CEP1 for the STRIKE tape and ABTAPE. If the record is for a missile, INTRFACE then reads in the target information blocks from the PLANTAPE. If the record is for a bomber, INTRFACE first reads in the plan information blocks to determine the number (KK) of scheduled weapons (i.e., the number of drop bomb and ASM target events), and then reads in KK target information blocks. If the record is for a tanker, INTRFACE reads in the plan information blocks from the PLANTAPE. For all weapon types, the associated events are sorted within sortie number on time of the event. If the user has specified that he does not want a STRIKE tape, INTRFACE skips the following process.

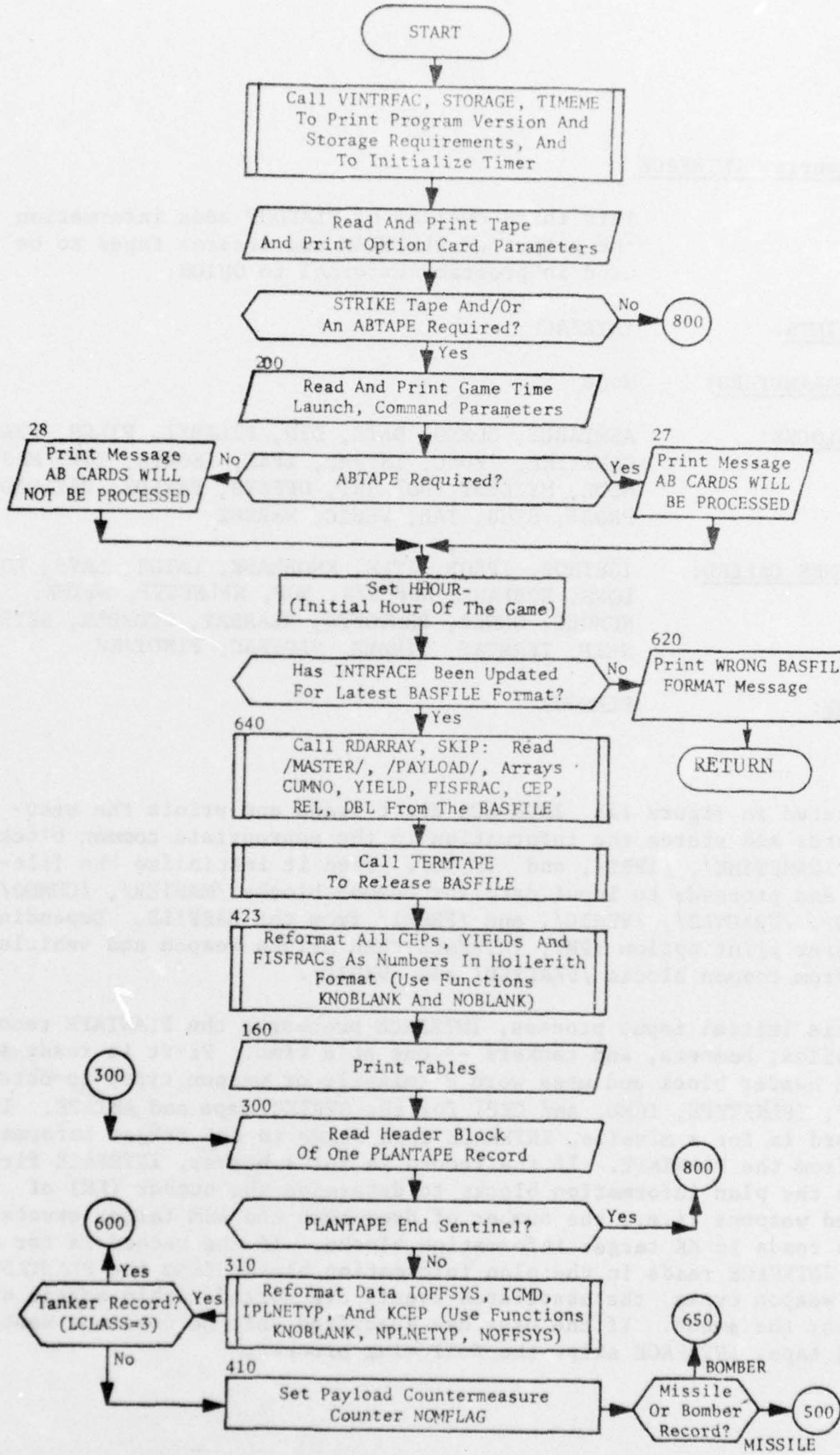


Figure 141. Subroutine INTRFACE (Part 1 of 5)

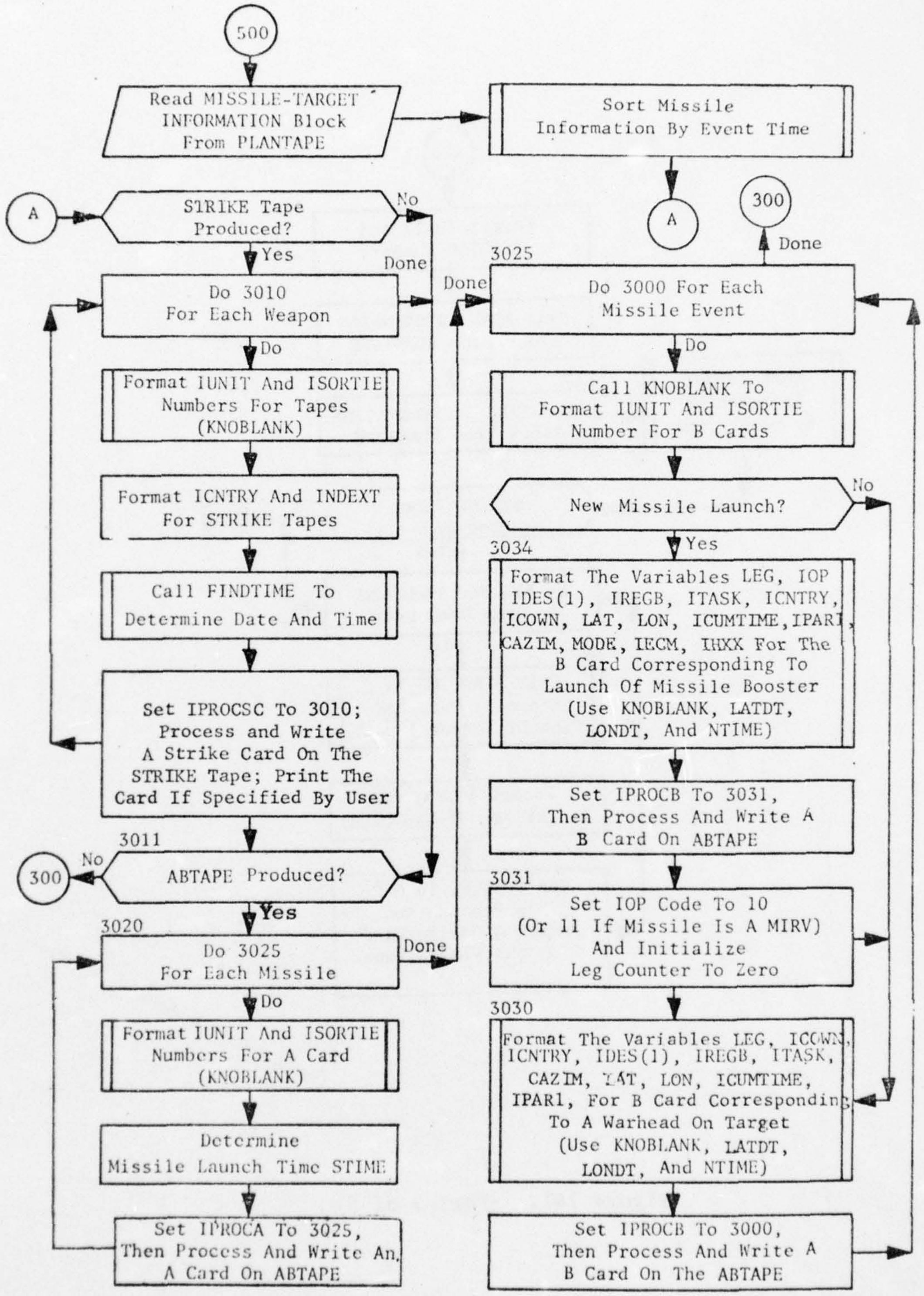


Figure 141. (Part 2 of 5)

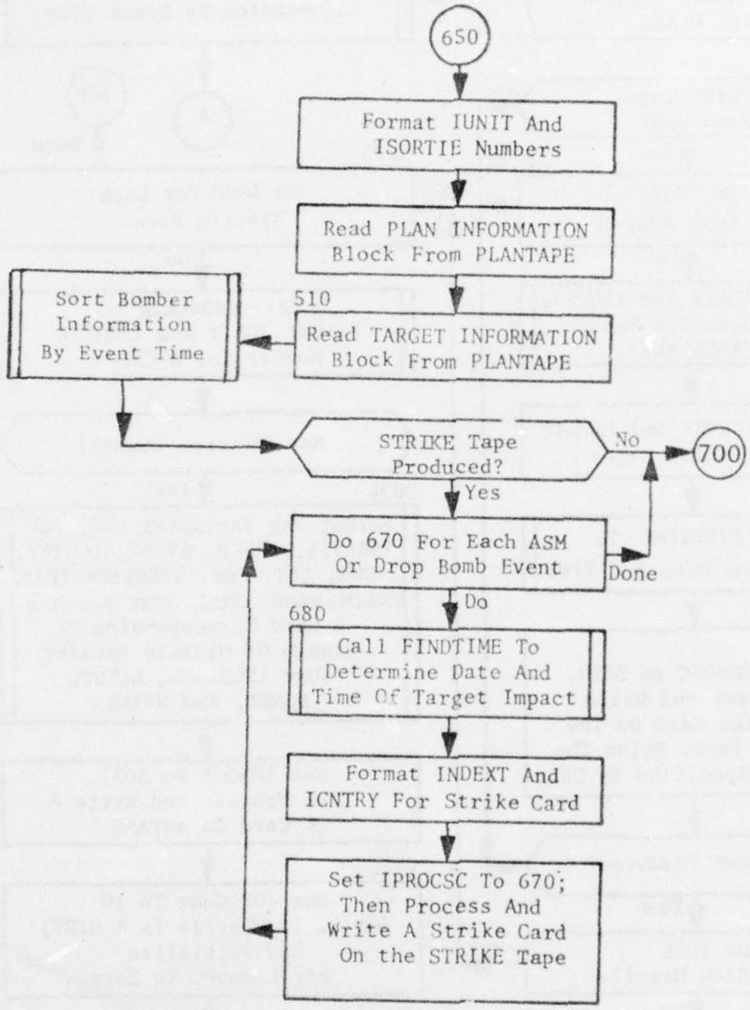


Figure 141. (Part 3 of 5)

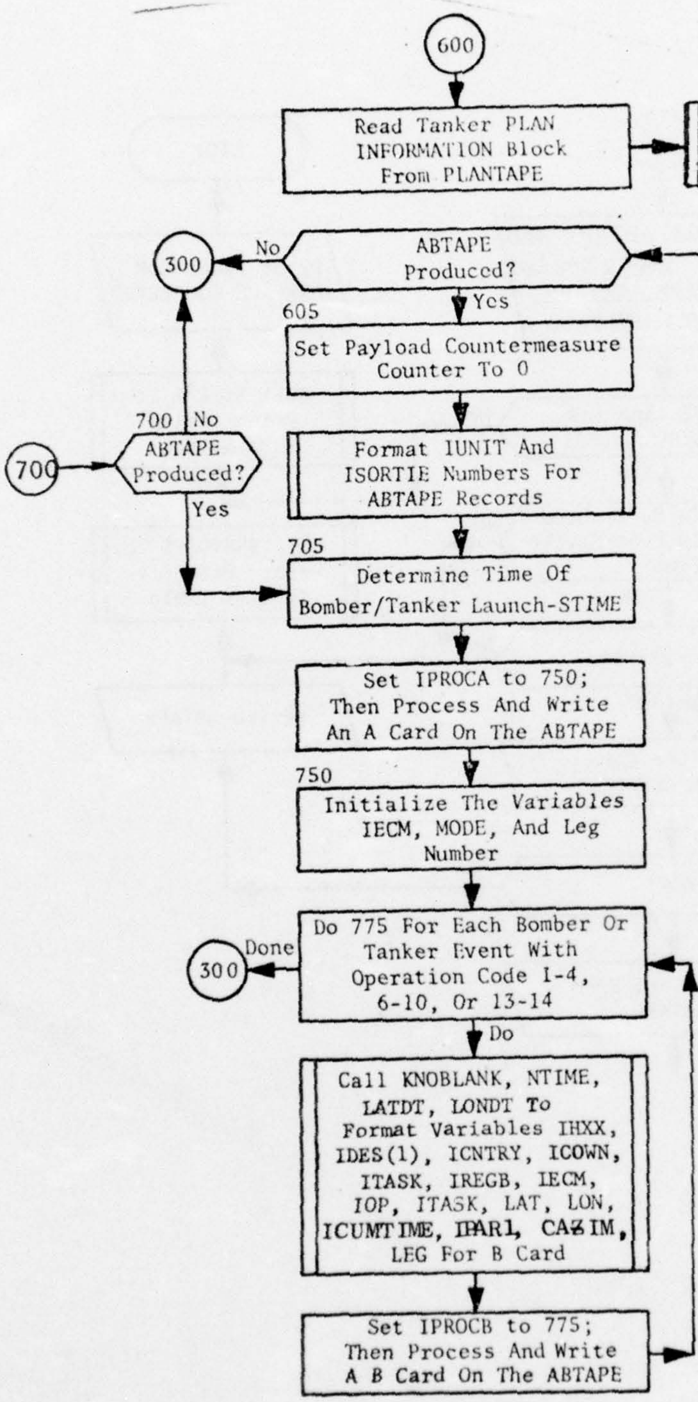


Figure 141. (Part 4 of 5)

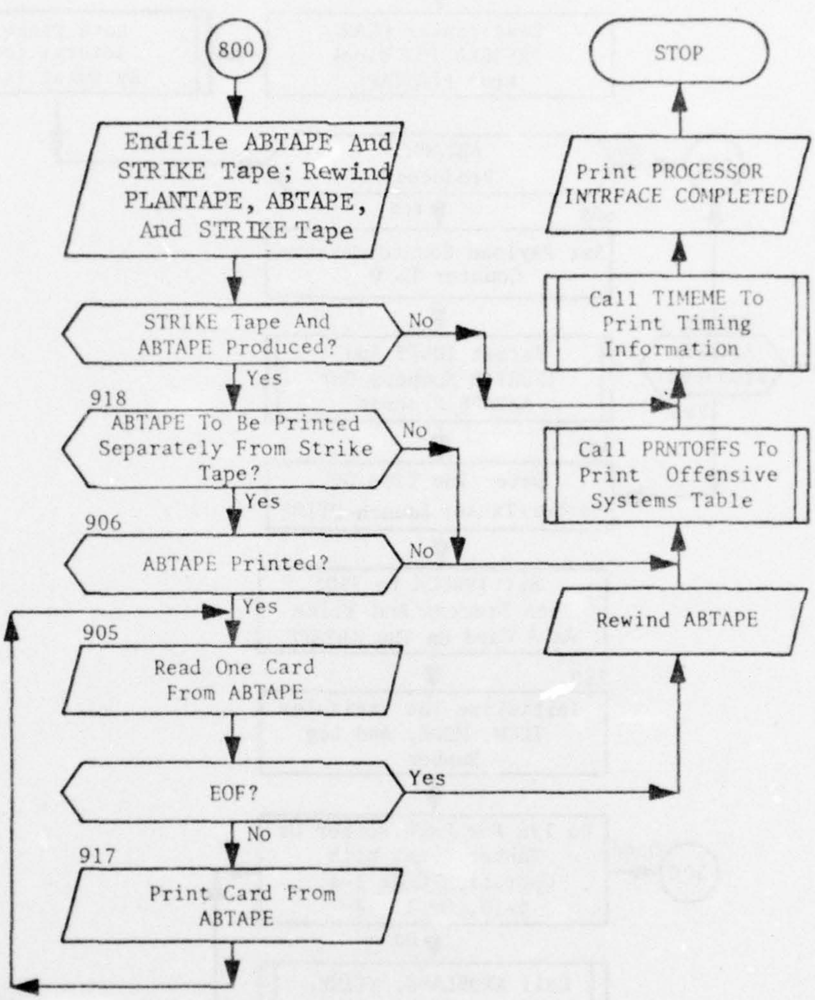


Figure 141. (Part 5 of 5)

Since INTRFACE produces strike cards only for events which are target hits by ASMs, bombs, or missiles, tanker records are ignored during the processing for the STRIKE tape. INTRFACE processing for the STRIKE tape falls into seven main categories. The first is conversion of many numbers into BCD code and replacement of leading blanks in these numbers by zeroes. Functions KNOBLANK and NOBLANK are used to accomplish this process. The second involves determining target impact time, TGT1TIME, in hours from the beginning of the game, and using common block /GAMETIME/ and subroutine FINDTIME to convert TGT1TIME into BCD coded data. The third is determination of target latitude and longitude in degrees and conversion of these numbers to a BCD coded word which contains the latitude or longitude in degrees, minutes, and seconds--North or South, or East or West. Function LATS (and entry LONS in LATS) are used to accomplish this process. The fourth is conversion of probabilities into percent by using function IPROB. The fifth is using IGETHOB to determine height of burst for the weapon. The sixth is using subroutine YLDFRAC to calculate equivalent yield and associated fission fraction if the record is for an MRV missile. The last is to compute the launch and back azimuths for missiles. When these seven categories of processing have been completed for missile and bomber records, INTRFACE writes a strike card record on the STRIKE tape and, depending on the user's option, also prints the card.

When all the strike cards corresponding to the PLANTAPE record have been produced, INTRFACE tests whether the user has specified that the ABTAPE is to be made. If not, INTRFACE skips the processing of "A" and "B" cards and returns to the branch where it reads in a PLANTAPE record. Otherwise, corresponding to each aircraft in the case of bomber or tanker record and to a booster in the case of a missile, INTRFACE processes information for the "A" card. Only two categories of processing are involved. The first is conversion of integers to BCD code as mentioned before. The second is calculation of bomber or tanker takeoff time or of missile launch time in hours and minutes from the beginning of the game; NTIME is used to convert the time from floating point hours to BCD hours and minutes. When this processing for the "A" card is complete, INTRFACE writes an "A" card record on the ABTAPE. If the user has specified that the ABTAPE is to be printed and that it should be printed together with the STRIKE tape, INTRFACE prints the "A" card. It then begins processing the "B" cards. It produces a "B" card corresponding to each missile launch, missile reentry vehicle on target, and to each bomber or tanker event which is a launch, refuel, dogleg, ASM launch, ASM on target, decoy release, decoy impact, bomb on target, recovery, or splash. Processing for this card falls into four main categories. The first is again conversion of intergers to BCD code using KNOBLANK and NOBLANK. The second is conversion of floating point numbers for latitude and longitude at the end of a leg into BCD codes; LATDT and entry LONDT in LATBT are used to perform this function.

The third is calculation of number of hours since beginning of game to the beginning of the route leg and conversion of this number into BCD code by NTIME. The last is determination of the target region associated with the missile reentry vehicle, ASM or bomb. At the completion of "B" card processing, INTRFACE writes the "B" card on the ABTAPE. Again, if the user specified printing of the ABTAPE together with the STRIKE tape, INTRFACE also prints the "B" card.

When all of the PLANTAPE records have been processed to produce the specified strike cards and/or "A" and "B" cards, INTRFACE writes end-of-file marks on the STRIKE and/or ABTAPE and tests to determine if the user specified a print of the ABTAPE separate from the STRIKE tape. If a separate print was specified, INTRFACE rewinds ABTAPE, reads it one card at a time, decodes it, and prints it; at the end of the print INTRFACE calls PRNTOFFS to print the offensive systems table for the PLANTAPE, and then prints INTRFACE timing information before stopping.

4.6.3.1 Subroutine FINDTIME

PURPOSE: To convert TGT1TIME from a floating point number of hours to a date and time in integer format where the date and time are computed from the base game time.

ENTRY POINTS: FINDTIME

FORMAL PARAMETERS: None

COMMON BLOCKS: DATE, GAMETIME

SUBROUTINES CALLED: KNOBLANK

CALLED BY: INTRFACE

Method:

TGT1TIME is the time in hours since the beginning of the game of a target hit. FINDTIME first converts TGT1TIME into integers IHOUR and MIN for minutes. XMIN, the floating point number of minutes, is further converted to ISEC for seconds where ISEC is a result of rounding off to the nearest integer. If ISEC equals 60, FINDTIME resets it to 0 and adds 1 to MIN. If MIN equals 60, FINDTIME resets 1 to 0 and adds 1 to IHOUR. Then if IHOUR is at least 24, FINDTIME alternately decreases IHOUR by 24 and increases IDAY by 1 until IHOUR is less than 24. (The initial values of IDAY, IMON, and IYEAR are the input values for the day, month, and year of the game.) Similarly, FINDTIME tests IDAY and if it is greater than 31 it alternately adds 1 to IMON and decreases IDAY by 31 until IDAY is less than 32. FINDTIME's final test is on IMON. If IMON is larger than 12, FINDTIME alternately decreases it by 12 and increases IYEAR by 1 until IMON is less than 13. If for some unanticipated reason IYEAR is larger than 99, FINDTIME resets it to 99 and prints an error message for ILLEGAL DATE.

Finally, FINDTIME codes IDAY, IHOUR and MIN into integer by performing the sum of 10000·IDAY, 100·IHOUR, and IMIN, to obtain IDATE. KNOBLANK is used to obtain DAHOMIMO, LDAY (now actually month), YEAR (actually now seconds) in BCD format.

Subroutine FINDTIME is illustrated in figure 142.

SECTION 5. PROGRAM TABLE

5.1 Purpose

Program TABLE provides an interface between QUICK and two external simulators used in RISOP/SIOP gaming; i.e., the Event Sequenced Program (ESP) and the Nuclear Exchange Model (NEMO).

Specifically, program TABLE reads either the INDEXDB tape produced by program INDEXER or the INMODDB tape produced by program DBMOD and records, in abbreviated form, selected data concerning weapon systems and targets described therein. The extracted data are written on an output tape, TABLTAPE. This program performs no other functions and is not required to operate the QUICK system. However, because it summarizes part of the indexed data base, it enables the user to review the data base before embarking on plan generation if he so chooses.

5.2 Input File

The sole input file is the INDEXDB tape produced by program INDEXER or the INMODDB tape produced by program DBMOD. The format of both these files is identical and is described as the output of INDEXER of the Weapon/Target Identification subsystem. For the remainder of this chapter, the input file is assumed to be the INDEXDB tape. The INMODDB tape replaces this file only if the user has used program DBMOD after program INDEXER to modify the indexed game data base.

The data extracted from the input file include descriptions of targets, missile and bomber launch bases, delivery vehicles (missiles and bombers), and weapon characteristics.

The sole user-input parameter to program TABLE specifies the attacking side for the current plan. Thus, TABLE is run once for each side.

5.3 Output File

The sole output file is the TABLTAPE. This tape is written as six data lists: Target list (F1TARGET), Vehicle Characteristics list (F1VEHIC), Weapon Characteristics list (F1WEAPON), Missile Base list (F1MIBASE), Offensive Recovery Base List (F1RECBS), and Bomber Base list (F1BASE). The lists are placed on the TABLTAPE in the above order. Each entry in each list is written as one 80-character BCD logical record. (The TABLTAPE consists of only one file; the lists are not separated into separate files.)

The information on the TABLTAPE is also printed on the standard output file to provide hard copy output of these lists. Figures 153 through 157 display the format of the 80-character records for each list.

COLUMNS	DESCRIPTION	REMARKS
1-8	Format and table name	FITARGET
9	Side	1 = BLUE 2 = RED
10-14	Line number	1 to 9999
15	Blank	
16-20	Target designator code (Desig)	2 Alpha 3 numeric
21-24	Blank	
25-31	Latitude	Degrees, minutes, seconds, S if south, blank if north
32-39	Longitude	Degrees, minutes, seconds, E if east, W if west
40-46	Blank	
47-54	Target name	8 characters only
55-59	Category code	5 numeric
60-61	Country code	2 Alpha
62-67	Major reference number	6 numeric
68	Blank	
69-70	Task	2 Alpha
71	Blank	
72-76	Index number (INDEXNO)	1-12,000 assigned by INDEXER
77	Blank	
78-80	Complex number	1-999 assigned by INDEXER

Figure 153. Target List (Program TABLE)

COLUMNS	DESCRIPTION	REMARKS
1-8	Format and table name	F1VEHIC
9	Side	1 = BLUE 2 = RED
10-14	Line number	1-9999
15	Card number	1
16-18	Blank	
19-20	Plane type	Type number = 1 to 99
21-55	Blank	
56-58	CEP Mode 1 or CEP Msl*	Hundreds of feet-0 to 999
59-61	Blank	
62-64	CEP Mode 4*	Hundreds of feet-0 to 999
65-67	Blank	
68-75	Vehicle type name	8 characters
76-80	Blank	

*Mode 1 (high altitude) and Mode 4 (low altitude) refer to bomber flight profiles. QUICK permits only one value of CEP for each type bomber. This assigned CEP is entered for both modes (cc 56-58 and 62-64)

Figure 154. Vehicle Characteristics List (Program TABLE)

COLUMNS	DESCRIPTION	REMARKS
1-8	Format and table name	FlWEAPON
9	Side	1 = BLUE 2 = RED
10-14	Line number	1 to 9999
15	Blank	
16-19	Weapon number	Weapon number = 1 to 50 (WHDTYPE)
20	Weapon type	0 = Bomb 1 = ASM 2 = Decoy
21-37	Blank	
38-43	Weapon yield	Kilotons
44-46	FISFRAC	000-100
47-80	Blank	

Figure 155. Weapon Characteristic List (Program TABLE)

COLUMNS	DESCRIPTION	REMARKS
1-6	Format and table name	F1BASE
7-8	Blank	
9	Side	1 = BLUE 2 = RED
10	Blank	
11-14	Base number (NUMBAS)	
15	Blank	
16-20	Unit identification number	QUICK index number INDEXNO (1-12000)
21	Blank	
22-28	Latitude	Degrees, minutes, seconds, S if south, blank if north
29-36	Longitude	Degrees, minutes, seconds, E if east, W if west
37	Blank	
38	Red launch command (Bomber function)	2 = LRA 3 = TAC 7 = None of the above
39	Blank	
40	Base functions (either home base or dispersal base)	X = yes: Blank or zero = no; Note: differentiation between a "home base" and a "dispersal base" is not made
41-43	Blank	
44	Tanker	**one Alpha character**
45-59	Blank	
60-67	Target name	**eight Alpha characters**
68-69	Blank	
70-71	Country Location	**two Alpha characters**
72-80	Blank	

Figure 157. Bomber Base List (Program TABLE)

COLUMNS	DESCRIPTION	REMARKS
1-8	Format and table name	FIRECBS
9	Side	1 = BLUE 2 = RED
10-14	Line number	1 to 9999
15	Blank	
16-20	Target designator code (DESIG)	2 Alpha 3 numeric
21-24	Blank	
25-31	Latitude	Degrees, minutes, seconds, S if south, blank if north
32-39	Longitude	Degrees, minutes, seconds, E if east, W if west
40-45	Target name	6 characters
46-55	WAC/BE	10 Alpha
56-60	Category code	5 numeric
61-62	Country code	2 Alpha
63-68	Major reference number	6 numeric
69-70	Task	2 Alpha
71-75	Index number (INDEXNO)	1-12,000
76	Blank	
77-80	Capacity	1-9999 Assigned by INDEXER

Figure 157.1 Offensive Recovery Base List (Program TABLE)

5.4 Concept of Operation

Program TABLE reads through the input data base one item at a time. All indexed items on the defending side are added to the target list (F1TARGET). The values of appropriate attributes are encoded into the 80-character entry for this list and written on the TABLTAPE directly.

The second and third lists on the TABLTAPE, Vehicle Characteristics list (F1VEHIC) and Weapon Characteristics list (F1WEAPON), are maintained in core during the operation of program TABLE. These lists are stored in common block /C111/ in arrays TABVEH and TABWEP, respectively.

The fourth list, the Missile Base list (F1MIBASE), is stored temporarily on a scratch file, OUTAP2. This file, produced by the QUICK filehandler, uses filehandler buffer number 2. The 80-character list entries are output as a 10-word data block. In the closing phases of TABLE processing, this scratch file is read and the data are transferred to the TABLTAPE.

The fifth list on the TABLTAPE, the Bomber Base list (F1BASE), is stored temporarily on a scratch file OUTAP3. This file uses filehandler buffer 4 and is used in the same manner as OUTAP2.

The sixth and final list on the TABLTAPE, the offensive Recovery Base List (F1RECBS) is also stored temporarily on a scratch file OUTAP4. This file uses filehandler buffer 8 and is used in the same manner as OUTAP2.

TABLE performs some simple error checking and data conversion. The number of entries in the vehicle and weapon tables is checked to prevent table overflow. In the vehicle table, the circular error probability (CEP) is converted from nautical miles (on the INDEXDB) to hundreds of feet (TABLTAPE). In the weapon table, the yield is converted from megatons to kilotons and the fission fraction is converted from a fraction (between 0.0 and 1.0) to hundredths (between 0 and 100).

Subroutine HELP is used to convert latitudes and longitudes. On the INDEXDB tape, these attributes are stored in QUICK system format. In this format, latitudes are expressed in degrees and fractions of degrees with north latitude positive and south latitude negative. Longitudes are expressed in degrees and fractions of degrees ascending in a westward direction from the Greenwich meridian in a range from 0.0 to 360.0 degrees. Subroutine HELP converts data from this format to the standard degrees/minutes/seconds/direction format.

5.5 Common Block Definition

Common blocks /DIRECTRY/ and /PROCESS/ of the data base handling package are used in processing the indexed data base (INDEXDB). In addition, /C111/, described below, is used for internal processing.

BLOCK	ARRAY	DESCRIPTION
C111	TABTAR(10)	Temporary storage for single entry in target list (F1TARGET)
	TABMIS(10)	Temporary storage for single entry in missile base list (F1MIBASE)
	TABVEH(800)	Vehicle characteristics list (F1VEHIC)
	TABBAS(10)	Temporary storage for single entry in bomber base list (F1BASE)
	TABWEP(700)	Weapon characteristics list (F1WEAPON)
	TABREC(10)	Temporary Storage for Single Entry in Recovery Base List (F1RECBS)

5.6 Program TABLE - Main Processor Routine

PURPOSE: This routine retrieves, reformats, and writes the information required for the TABLTAPE.

ENTRY POINTS: TABLE

FORMAL PARAMETERS: None

COMMON BLOCKS: DIRECTRY, EDITERM, ITP, MYIDENT, NOPRINT, PROCESS, WAROUT

SUBROUTINES CALLED: HELP, INITAPE, INITEDIT, INPITEM, ITLE, NEXTITEM, RDARRAY, SETREAD, SETWRITE, TERMTAPE, WRARRAY, GLOG, SLOG

CALLED BY: Operating System; this is a main program

Method:

This routine is the main processor. The processing is quite straightforward. The input data base is investigated item by item. A series of checks determines if the item is a target, launch base, or weapon. If not, the item is ignored. If the item is one of these, control transfers to a part of this routine which reformats the appropriate attribute values into the form required on the TABLTAPE.

Three local arrays are used in this process:

VEH(200) - A logical array; set true if vehicle type has already been processed to vehicle table

NYLD(50) - Yield in kilotons for each warhead

NFFRAC(50) - Fission fraction in hundredths for each warhead.

Figure 158 is a flowchart of this routine.

In the series of statements preceding statement 29 several calls on utility subroutine ITLE are made. These calls look up the index of various attributes in the data base directory (array ATTNAME in common /DIRECTRY/). These indices are used to retrieve the attribute values from the VALUE array in common /PROCESS/. This mode of operation obviates the need for processing the TABLE source code with utility program DECLARES.

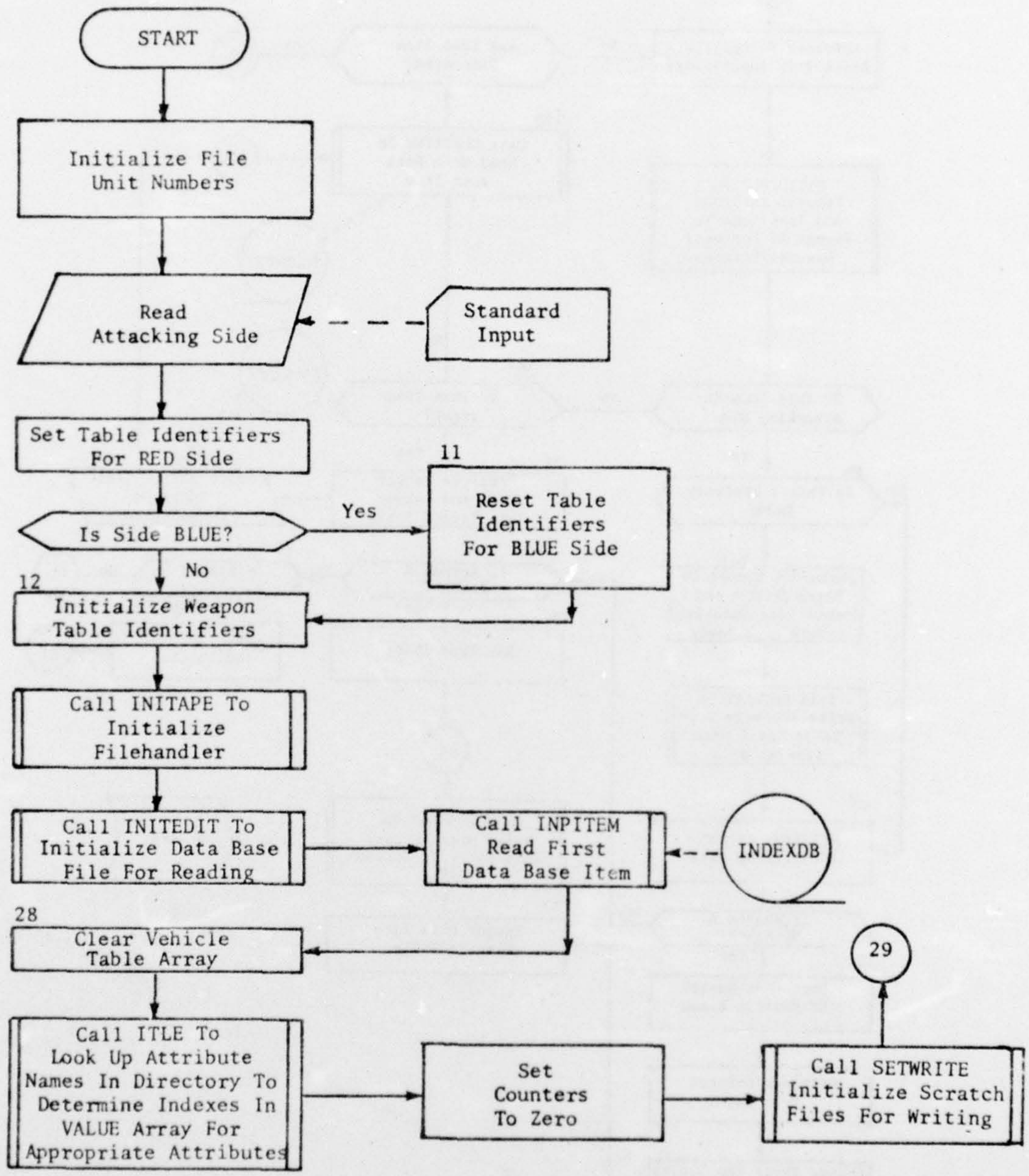


Figure 158. Program TABLE (Part 1 of 5)
Part I: Initialization

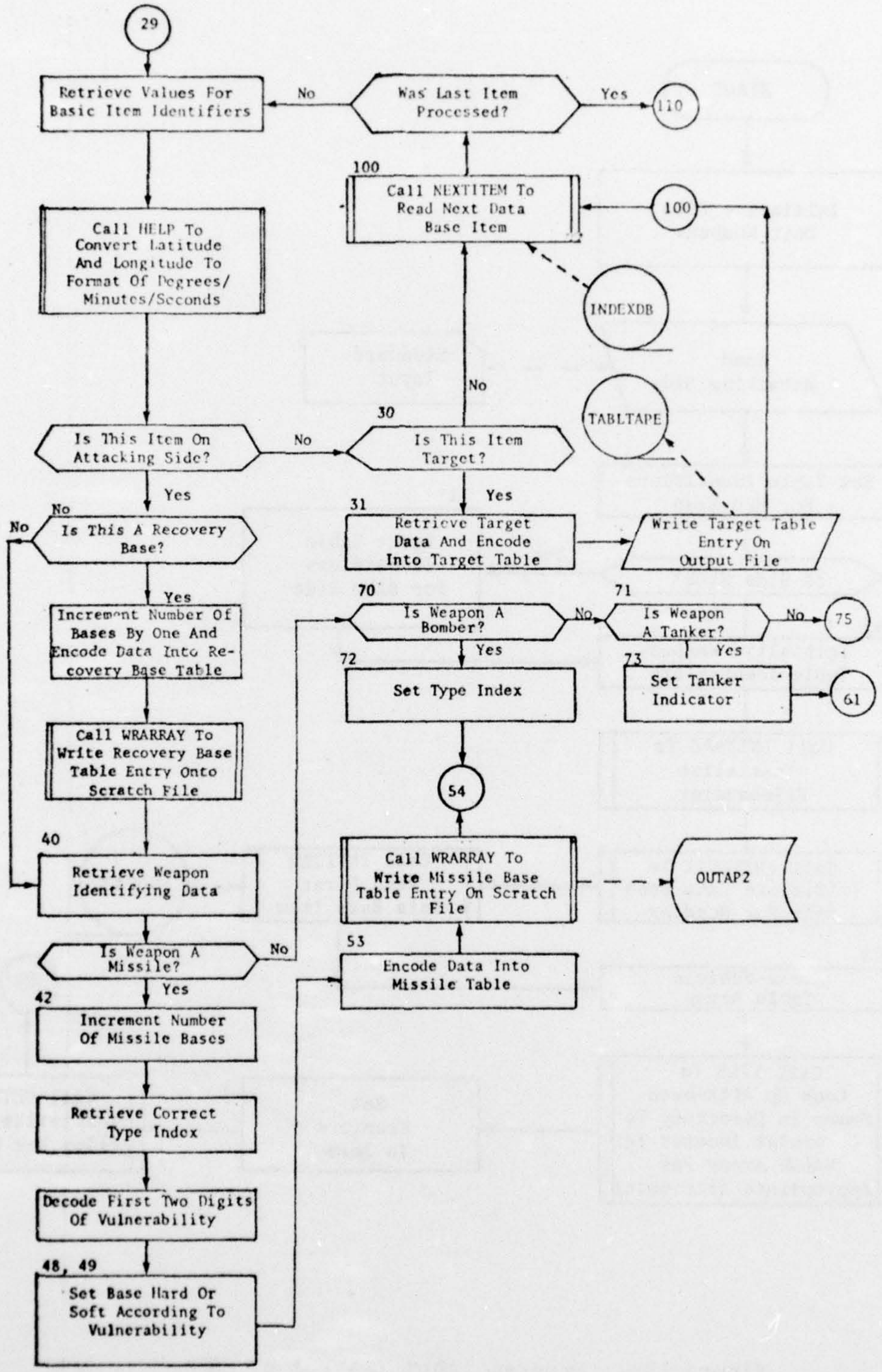


Figure 158. (Part 2 of 5)
Part II: Basic Item Processing

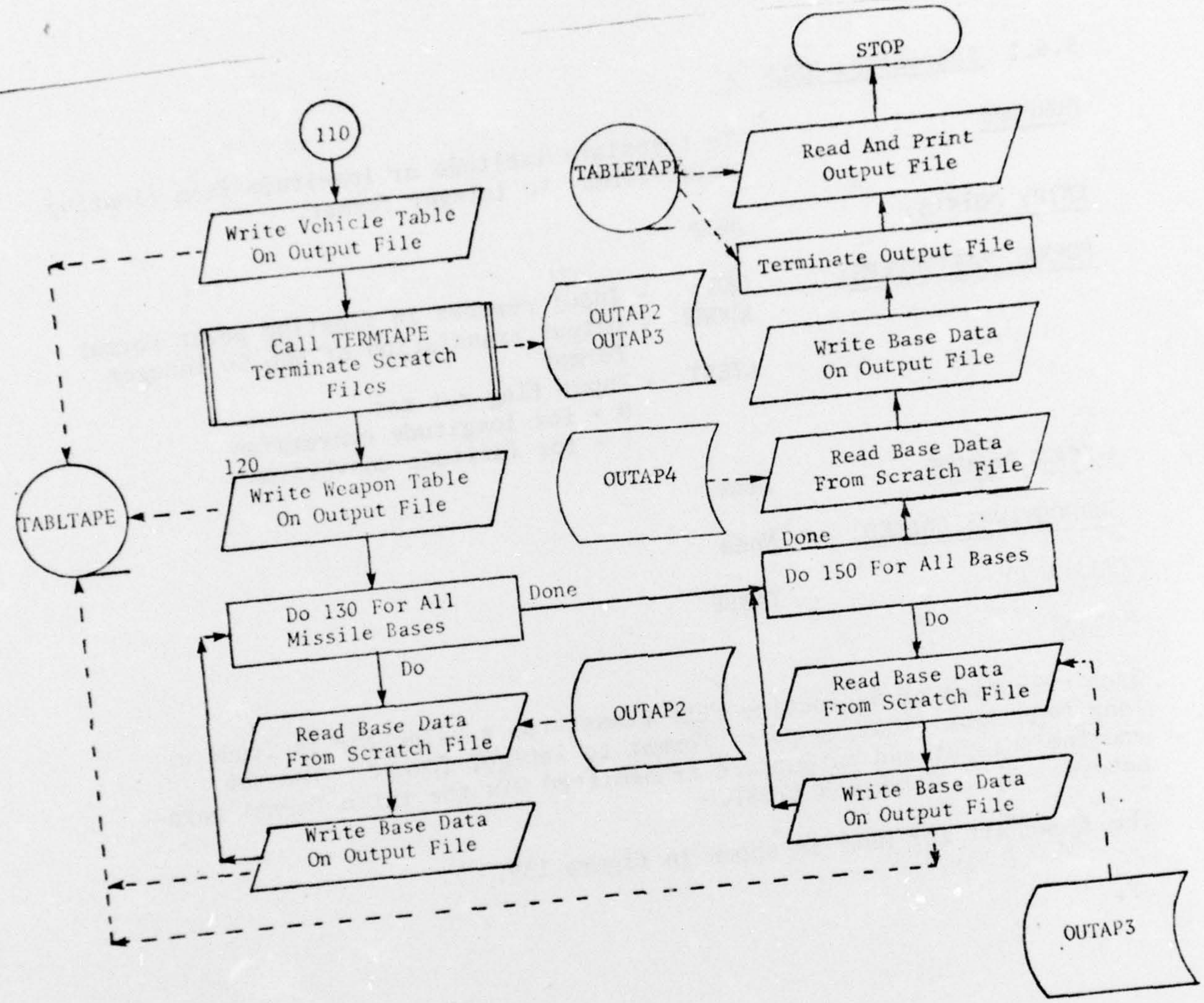


Figure 158. (Part 5 of 5)
Part V: Termination Processing

CH-2

5.6.1 Subroutine HELP

PURPOSE: To translate latitude or longitude from floating point format to integer format.

ENTRY POINTS: HELP

FORMAL PARAMETERS:

DEG	-	Input degrees in floating point format
KDONE	-	Output translation of DEG to integer format
LTEST	-	Input flag set to: 0 - for longitude conversion 1 - for latitude conversion

COMMON BLOCKS: None

SUBROUTINES CALLED: None

CALLED BY: TABLE

Method:

Each execution of subroutine HELP translates a value for latitude or longitude from floating point format to integer format. The subroutine's input and output are transmitted via the three formal parameters: DEG, KDONE and LTEST.

The flowchart for HELP is shown in figure 159.

Vol 4
AO38893